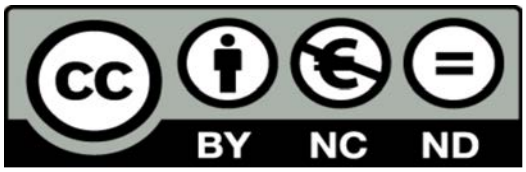
UNIVERSIDAD DE LA RIOJA

TESIS DOCTORAL

Título
Diseño de modelos de fuerzas aplicando técnicas de Machine Learning. Método de Encke híbrido
Autor/es
Hans Mauricio Carrillo Hernández
Director/es
Juan Félix San Juan Díaz y Iván Luis Pérez Barrón
Facultad
Facultad de Ciencia y Tecnología
Titulación
Departamento
Matemáticas y Computación
Curso Académico
2023-2024



publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Tesis Doctoral

Diseño de modelos de fuerzas aplicando técnicas de Machine Learning. Método de Encke híbrido

Hans Mauricio Carrillo Hernández

Universidad de La Rioja
Departamento de Matemáticas y Computación
Logroño, España
2024

Diseño de modelos de fuerzas aplicando técnicas de Machine Learning. Método de Encke híbrido

Hans Mauricio Carrillo Hernández

Memoria presentada como requisito parcial para optar al título de:
Doctor en Matemáticas y Computación

Director:
Ph.D Juan Félix San Juan Díaz

Codirector:
Ph.D Iván Luis Pérez Barrón

Línea de Investigación:

Métodos formales en ingeniería del software y cálculo científico en matemáticas.

Centro de Computación Científica e Innovación Tecnológica SCoTIC.

Universidad de La Rioja
Departamento de Matemáticas y Computación
Logroño, España
2024

– A mi madre, quien pese a todas las dificultades que tuvo que pasar, siempre estuvo brindándome su apoyo incondicional. A mi pareja, mis hermanas, mis dos rebeldes sobrinos y al recuerdo de mis abuelos maternos, todos ellos han estado presentes en cada una de las frases que he escrito en esta memoria.

Agradecimientos

Mi más profundo agradecimiento a mi director de tesis, el Dr. Juan Félix San Juan, director del Centro de Computación Científica e Innovación Tecnológica de la universidad de la Rioja, una persona inteligente, con valores admirables y una dedicación extraordinaria a su trabajo, su conocimiento experto y consejo crítico han sido insustituibles en el desarrollo de esta investigación. De igual manera, deseo expresar mi reconocimiento a mi codirector de tesis, Iván Pérez Barrón, que con su meticulosa atención al detalle, sus correcciones y su apoyo, han enriquecido enormemente mi investigación. Quiero extender además mi agradecimiento a mis compañeros del Centro de Computación Científica. Edna Viviana Segura, que siempre estuvo dispuesta a escucharme, sus valiosas reflexiones y puntos de vista, me permitieron avanzar en mis momentos de confusión. A Rosario López y Manuel Higueras por su invaluable apoyo y colaboración a lo largo del doctorado. Gracias a todos ellos he podido culminar con esta investigación y elaborar esta memoria.

Esta tesis ha sido inicialmente financiada con fondos del Centre National D'Etudes Spatiales - Centro Nacional de Estudios Espaciales (CNES) (proyecto R&T R-S20/BS-0005-059) y el Fondo Europeo de Desarrollo Regional (FEDER, proyecto del PID2021-123219OB-I00). Posteriormente, se ha desarrollado en mi condición de beneficiario desde el 1 de septiembre de 2021, de un contrato predoctoral para la formación de personal investigador, aprobado en la resolución 970 de 29 de julio de 2021, financiado por la Universidad de La Rioja y por la Comunidad Autónoma de La Rioja.

Contenido

Lista de Figuras

Lista de Tablas

Introducción

Hoy en día los satélites se han convertido en una herramienta vital para el progreso de la humanidad. Desde la comunicación hasta la navegación, pasando por la observación terrestre y la meteorología, los satélites están presentes en casi todos los aspectos de nuestra vida [1]. Uno de los servicios más destacados que prestan los satélites es la comunicación. Empresas como OneWeb o SpaceX están proporcionando conexión de banda ancha mediante mega-constelaciones formadas por miles de satélites (ver [2], pag 144). Gracias a ellos, se puede acceder a Internet desde cualquier parte del planeta, incluso en áreas remotas donde la infra-estructura terrestre es limitada o inexistente. Además, los satélites son indispensables para la navegación precisa. Los sistemas de posicionamiento global (GPS, GNSS, GLONASS [3] , BeiDou [4] y Galileo [5]) permiten planificar rutas, localizar direcciones y monitorizar todo tipo de vehículos, ya sean aéreos, marítimos o terrestres, en tiempo real. En el ámbito de la observación de la Tierra, los satélites proporcionan imágenes de alta resolución que se utilizan en agricultura para monitorizar cultivos, prevenir desastres naturales como incendios forestales e inundaciones y realizar estudios ambientales para la conservación de la biodiversidad y la gestión de recursos naturales [6]. Asimismo, los satélites desempeñan un papel crucial en la predicción del clima, permitiendo a los meteorólogos monitorizar patrones climáticos, prever tormentas y huracanes, y proporcionar alertas tempranas sobre condiciones climáticas extremas [7].

Sin embargo, para garantizar que los satélites puedan prestar sus servicios es fundamental asegurar un entorno espacial seguro. Las principales causas que pueden alterar este entorno son la meteorología espacial, conocida como *space weather* en inglés, y los restos de basura espacial, denominados *space debris* en inglés. La meteorología espacial se refiere a las condiciones ambientales que se dan en el espacio, las cuales están principalmente influenciadas por la actividad solar. Por otro lado, en la segunda causa se considera que la trayectoria de un satélite puede verse afectada por las de otros satélites o residuos espaciales, lo que podría llegar a provocar una colisión [8].

Recientemente, el problema que genera la gestión de la basura espacial es abordado conceptualmente como el de un *ecosistema* que requiere protección. Similar a un ecosistema terrestre, cada objeto en órbita, ya sea un satélite activo o un resto de basura espacial, desempeña un papel único y contribuye a la estabilidad del conjunto. Sin embargo, la proliferación de objetos en el espacio plantea amenazas significativas para la sostenibilidad y la seguridad del ecosistema [9].

Según los datos de la Agencia Espacial Europea, European Space Agency en inglés (ESA), hasta diciembre de 2023,[1] de los 16 990 satélites lanzados desde 1957, 11 500 permanecen en órbita, de los cuales, aproximadamente 9000 están en funcionamiento. Durante este tiempo,

[1] `https://www.esa.int/Space_Safety/Space_Debris/Space_debris_by_the_numbers`

se estima que se han generado alrededor de 36 500 residuos espaciales con una tamaño mayor a 10 cm, 1 000 000 con una tamaño entre 1 y 10 cm, y se estima que más de 130 000 000 con una tamaño de entre 1 mm y 1 cm. Dado que la eliminación de los residuos espaciales no es técnica ni económicamente viable, por el momento, el esfuerzo por mantener el espacio seguro se centra en dos tareas principales: evitar la creación de nuevos residuos espaciales y realizar un seguimiento de los residuos ya existentes con el fin de poder catalogarlos. Para evitar la generación de residuos espaciales, el Comité de Coordinación Inter-institucional de Residuos, Inter-Agency Space Debris Coordination Committee en inglés, ha estado publicando regularmente directrices desde el año 2002. La más reciente, publicada en junio de 2021[2], proporciona directrices para prevenir la generación de residuos por colisiones en órbita de los satélites o durante el proceso de lanzamiento. Además, insta a las agencias miembro a retirar los satélites que han cumplido su misión, a regiones orbitales determinadas y a limitar la liberación de residuos durante todas las operaciones normales del satélite.

El proceso de creación de un catálogo de basura espacial implica varias etapas. En primer lugar, se utilizan sistemas de vigilancia tanto terrestres como espaciales para detectar y rastrear objetos en órbita alrededor de la Tierra. Una vez detectados, se emplean técnicas de propagación y determinación orbital para identificar y caracterizar los objetos [10]. La información recopilada se organiza en un catálogo, que incluye datos detallados sobre cada objeto, como su designación, características orbitales y fecha de detección. Este catálogo se actualiza regularmente con nueva información y se realiza un seguimiento continuo de los objetos para mantener su precisión y relevancia a lo largo del tiempo.

La propagación orbital es una de las tareas frecuentes que se realiza durante la creación y el uso de catálogos de basura espacial. Un *propagado orbital* es una implementación, en un lenguaje de programación, de una solución del sistema dinámico que describe el movimiento de un satélite o de un resto de basura espacial alrededor de la Tierra. Es decir, un propagador orbital proporciona la posición y la velocidad de un objeto en el espacio en un tiempo dado t_f a partir de su posición y su velocidad en un tiempo inicial t_i. Es importante destacar que estas herramientas son también fundamentales para el análisis y el diseño de misiones espaciales.

Desde el punto de vista clásico, existen tres técnicas que permiten construir la solución de este sistema dinámico [11, 12, 13]. Las dos primeras, son conocidas como *técnicas generales de perturbación* [14, 15, 16] y *técnicas especiales de perturbación* [17, 18, 19]. Las técnicas generales, también llamadas *teorías analíticas*, están basadas en la integración analítica de las ecuaciones del movimiento utilizando la teoría de perturbaciones. Esta teoría proporciona soluciones analíticas aproximadas basadas en desarrollos en serie, las cuales son válidas para cualquier conjunto de condiciones iniciales. Es importante destacar que las teorías analíticas consideran un modelo básico de fuerzas que sólo capturan las características esenciales del problema, y que las soluciones sólo incluyen los primeros términos del desarrollo en serie. Así, esta técnica proporciona soluciones poco precisas pero eficientes desde el punto de vista computacional. Por otro lado, las técnicas especiales están basadas en la integración numérica de las ecuaciones del movimiento, lo que les permite modelar cualquier tipo de fuerza que afecte al movimiento del objeto. Los propagadores desarrollados a partir de esta técnica

[2] `https://orbitaldebris.jsc.nasa.gov/library/iadc-space-debris-guidelines-revision-3.pdf`

son precisos; sin embargo, debido a su dependencia del paso de integración utilizado, no son tan eficientes desde el punto de vista computacional. La tercera técnica, conocida como *teoría semi-analítica* [20, 21, 22], combina elementos de las técnicas generales y especiales. Mediante la teoría de perturbaciones, se construye un cambio de variables que desacopla el movimiento de los términos de corto período de los de largo período. Esto hace que sólo los términos de largo período permanezcan en las ecuaciones del movimiento, lo que les permite tomar pasos de integración más largos al ser integrados numéricamente. Mientras tanto, los términos de corto período quedan retenidos en las ecuaciones del movimiento. Esto permite considerar modelos de fuerzas tan completos como en el caso de las teorías especiales, lo que resulta en una solución precisa. Además, la utilización de pasos de integración más grandes le confiere a la teoría semi-analítica una eficiencia computacional mayor que la de las técnicas especiales. Podríamos concluir que las soluciones del sistema dinámico básicamente dependen de dos factores: las fuerzas que perturban la trayectoria del objeto y la técnica de integración empleada. Por lo tanto, el número de soluciones, y por ende el de propagadores que existen, es muy alto para enumerarlos. Sin embargo, esta variedad de propagadores es importante debido a que los escenarios en los que se deben emplear son muy variados y no siempre requieren precisión, pero si velocidad de cómputo.

Una vez seleccionado el propagador, es decir, la solución que implementa, cualquier tipo de mejora implica modificar de alguna manera la solución y, por lo tanto, su implementación, lo que afectará directamente a la validación del propagador. No obstante, en [23, 24, 25] se propone una cuarta técnica conocida como *metodología de propagación híbrida*. Al igual que las teorías semi-analíticas combinan las técnicas generales y especiales, la metodología híbrida combina cualquiera de las tres teorías anteriores con métodos de predicción basados en técnicas estadísticas o métodos de Inteligencia Artificial. En esta cuarta aproximación, los métodos predictivos se utilizan para incrementar el modelo de fuerzas que perturban la trayectoria del objeto, así como la precisión y la eficiencia computacional de los métodos de integración empleados, e incluso para abordar las incertidumbres inherentes a los modelos de fuerza a partir de observaciones precisas, tanto provenientes de observaciones reales como generadas por propagadores precisos. Es importante destacar que esta técnica es no invasiva, ya que el módulo predictivo es independiente del propagador inicial, por lo que no se modifica su implementación.

Esta tesis tiene como objetivo el desarrollo de un nuevo propagador híbrido de tipo Encke, que utilizará el propagador semianalítico SGP4 como intermediario y se denominará $\text{HEncke}_{\text{SGP4}}$. En lugar de integrar las diferencias, este nuevo propagador las predecirá mediante el uso de redes neuronales multicapa. El propagador $\text{HEncke}_{\text{SGP4}}$ estará especialmente adaptado para objetos que orbitan la región de órbita de altitud media, Mean Earth Orbits (MEO) por su denominación en inglés.

Se considera SGP4 debido a que es el propagador utilizado por el Comando de Defensa Aeroespacial de América del Norte (NORAD, por sus siglas en inglés [3]) para generar el único catálogo de residuos espaciales disponible al público general bajo ciertas condiciones. Este catálogo se distribuye en un formato llamado Two-Line Element (TLE) y registra la trayectoria de aproximadamente 31 300 residuos espaciales. Por otro lado, la parte híbrida

[3] Accesible en la dirección web: `https://celestrak.org/NORAD/elements/`

del propagador implementa el modelo de redes neuronales debido a su capacidad de generalización, lo que permite que un único modelo pueda predecir las diferencias entre SGP4 y el modelo de perturbaciones completo para cualquier TLE de la región MEO.

Esta memoria se ha dividido en cinco capítulos e incluye dos anexos. En el primer capítulo se realiza una breve introducción al problema de la propagación orbital. El segundo capítulo presenta algunos conceptos básicos de la Inteligencia Artificial, haciendo especialmente énfasis en las redes neuronales. A continuación, en el tercer capítulo se expone la metodología híbrida, acompañada de un estudio destinado a comprender el comportamiento del propagador SGP4 en la región MEO del espacio. El cuarto capítulo describe la construcción de un propagador híbrido de tipo Encke basado en SGP4. Finalmente, en el quinto capítulo se exponen las conclusiones obtenidas y se proponen varias líneas de investigación futuras.

1. Aproximación al movimiento de los satélites artificiales

Hace aproximadamente 400 años, el astrónomo y matemático aleman Johanes Kepler enunció las leyes que rigen el movimiento de los planetas en un potencial gravitatorio $V(\boldsymbol{r}) = 1/\boldsymbol{r}$. Por simples que parezcan, estas leyes también permiten describir el movimiento tanto de los satelites activos como la de los restos de escombros espaciales, en general objetos residentes en el espacio (por su denominación en inglés, Resident Space Objects, RSO) con una precisión razonable. Esto es debido a que la fuerza ejercida por la Tierra sobre el satélite, supera en varios órdenes de magnitud al resto de las fuerzas que actúan sobre él. No obstante, el análisis y el diseño de las misiones espaciales así como las operaciones relacionadas con la seguridad y vigilancia del espacio requieren modelos más complejos, que describan con mayor precisión la trayectoria del satélite. Estos modelos incluyen las fuerzas que no han sido consideradas en la aproximación de Kepler.

En este capítulo se introducen las características de la trayectoria y las ecuaciones del movimiento del modelo propuesto por Kepler a partir de la segunda ley de Newton. A continuación, se describen algunas de las fuerzas que pueden modificar la trayectoria de un satélite como son la no esfericidad de la Tierra, la atracción gravitacional del Sol y de la Luna, el rozamiento atmosférico y la presión de radiación solar. Para finalizar, se formulan las ecuaciones del movimiento del satélite utilizando los métodos de Cowell y de Encke.

1.1. El modelo de Kepler

Si se considera, primero, que la forma de la Tierra es perfectamente esférica y su distribución de masa uniforme y, segundo, que la masa de cualquier satélite se puede considerar despreciable en comparación con la de la Tierra, el satélite experimenta una aceleración producida por la fuerza de atracción gravitatoria terrestre de acuerdo a la segunda ley de Newton:

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r}, \tag{1-1}$$

donde $\boldsymbol{r}$ representa la posición del satélite con respecto al centro de la Tierra. El parámetro físico $\mu = GM$ es el coeficiente gravitacional. G es la constante de gravitación universal y toma el valor de $(6.67259 \pm 0.00085) \cdot 10^{-11}\,\mathrm{m^3kg^{-1}s^{-2}}$, mientras que M es la masa de la Tierra y su valor es de $5.9722 \cdot 10^{24}\mathrm{kg}$ [26].

Una de las consecuencias del movimiento en un potencial central, como es el caso del problema de Kepler, es que el movimiento es siempre plano. De hecho, si se calcula el producto vectorial entre $\boldsymbol{r}$ y la ecuación (1-1), se obtiene

$$\boldsymbol{r} \times \ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}(\boldsymbol{r} \times \boldsymbol{r}) = 0. \tag{1-2}$$

Esto quiere decir que la trayectoria del satélite está confinada dentro del plano orbital. Además, si a la ecuación (1-2) le añadimos $\dot{\boldsymbol{r}} \times \dot{\boldsymbol{r}}$, lo cual es trivialmente igual a cero, se puede reescribir como

$$\boldsymbol{r} \times \ddot{\boldsymbol{r}} = \boldsymbol{r} \times \ddot{\boldsymbol{r}} + \dot{\boldsymbol{r}} \times \dot{\boldsymbol{r}} = \frac{d}{dt}(\boldsymbol{r} \times \dot{\boldsymbol{r}}). \tag{1-3}$$

Por lo tanto, se obtiene que la derivada respecto del tiempo de $\boldsymbol{r} \times \dot{\boldsymbol{r}}$ es igual a cero, lo que implica que esta cantidad es constante

$$\boldsymbol{r} \times \dot{\boldsymbol{r}} = \boldsymbol{h}. \tag{1-4}$$

donde $\boldsymbol{h}$ es una constante de integración. El vector $\boldsymbol{h}$ recibe el nombre de *momento angular*, y tanto él como su módulo $h = |\boldsymbol{h}|$ permanecen constantes a lo largo de la trayectoria del satélite (segunda Ley de Kepler, la línea que une un planeta y el Sol barre áreas iguales en tiempos iguales). h es también conocida como la *velocidad areolar*.

1.1.1. Forma de la órbita

El modelo propuesto por Kepler posee una segunda constante de integración, la cual permite probar que las trayectorias de los satélites son cónicas. Esta constante recibe el nombre de *vector de Laplace-Runge-Lenz* o también es llamada *vector de excentricidad*[1]:

$$\boldsymbol{A} = \dot{\boldsymbol{r}} \times \boldsymbol{h} - \frac{\mu}{r}\boldsymbol{r}, \tag{1-5}$$

donde $r = |\boldsymbol{r}|$. Como se puede observar, el vector $\boldsymbol{A}$ también se encuentra en el plano orbital.

A continuación, si se multiplica escalarmente la ecuación (1-5) por el vector de posición se obtiene

$$\boldsymbol{A} \cdot \boldsymbol{r} = (\dot{\boldsymbol{r}} \times \boldsymbol{h}) \cdot \boldsymbol{r} - \frac{\mu}{r}\boldsymbol{r} \cdot \boldsymbol{r} = -(\boldsymbol{h} \times \dot{\boldsymbol{r}}) \cdot \boldsymbol{r} - \frac{\mu}{r}\boldsymbol{r} \cdot \boldsymbol{r}, \tag{1-6}$$

Teniendo en cuenta que $(\boldsymbol{h} \times \dot{\boldsymbol{r}}) \cdot \boldsymbol{r} = (\boldsymbol{r} \times \dot{\boldsymbol{r}}) \cdot \boldsymbol{h} = \boldsymbol{h} \cdot \boldsymbol{h} = h^2$, $\boldsymbol{r} \cdot \boldsymbol{r} = r^2$, e introduciendo el ángulo ν como el ángulo entre $\boldsymbol{A}$ y el vector posición $\boldsymbol{r}$, conocido como *anomalía verdadera*, se obtiene

$$h^2 = \mu r + Ar\cos\nu. \tag{1-7}$$

donde $A = |\boldsymbol{A}|$. Además, si se introducen las cantidades auxiliares $p = h^2/\mu$ y $e = A/\mu$ en la ecuación (1-7), se obtiene la ecuación que define la trayectoria del satélite en el plano orbital:

$$r = \frac{p}{1 + e\cos\nu}. \tag{1-8}$$

[1] La deducción del vector de Laplace se puede consultar en la sección 2.9 de [27] o en la página 102 de [28].

Las cantidades e y p reciben los nombres de *excentricidad* y *semilatus rectum*, respectivamente. e es una constante positiva e indica la forma de la órbita, es decir, su redondez o achatamiento. El valor de p de las secciones cónicas resulta ser el radio de curvatura en los vértices del eje focal. Por otro lado, es fácil comprobar que la posición del satélite varía entre un valor mínimo de

$$r_{min} = \frac{p}{1+e}, \tag{1-9}$$

que se alcanza cuando $\nu = 0$, y un valor máximo de

$$r_{max} = \begin{cases} \frac{p}{1-e} & \text{si } 0 \leq e \leq 1, \\ \infty & \text{si } 1 \leq e. \end{cases} \tag{1-10}$$

A estos puntos se les conoce como el *perigeo* y el *apogeo* de la órbita, respectivamente, mientras que a su conexión se le conoce como la *línea de los ápsides*. En el caso de que la órbita presente una distancia con el apogeo finita, se define el *semieje mayor* de la órbita a como el valor medio entre las distancias mínima y máxima:

$$a = \frac{1}{2}(r_{min} + r_{max}) = \frac{h^2}{\mu(1-e^2)}. \tag{1-11}$$

Para finalizar, es importante destacar que la ecuación (1-8) representa a una sección cónica en coordenadas polares. De modo que, si la excentricidad toma un valor menor que 1 obtenemos una trayectoria elíptica, para un valor igual a 1 la trayectoria se convierte en parabólica, y si toma un valor mayor que 1 se trata de una trayectoria hiperbólica. En este trabajo sólo se abordará el movimiento elíptico (primera Ley de Kepler, las órbitas de los planetas son elipses con el Sol en uno de sus focos).

1.1.2. Dependencia temporal del movimiento. La ecuación de Kepler

Aunque es muy importante conocer cuáles son las posibles trayectorias que permiten describir el modelo de Kepler, ésto no es suficiente para determinar la posición del satélite en un determinado instante de tiempo. Como se podrá comprobar, es la ecuación de Kepler la que proporciona la solución temporal del problema. Con este propósito, se va a introducir una variable auxiliar E, denominada *anomalía excéntrica*.

Desde el punto de vista geométrico, esta variable representa el ángulo que forma la proyección del satélite sobre la circunferencia principal y el eje de la elipse (ver figura **1-1**). Las coordenadas x e y representan la posición del satélite en el plano orbital y la relacionan con ν y E de la siguiente manera:

$$\begin{aligned} x &= r\cos\nu = a(\cos E - e), \\ y &= r\,\mathrm{sen}\,\nu = a\,\mathrm{sen}\,E\sqrt{1-e^2}. \end{aligned} \tag{1-12}$$

De la ecuación (1-12) es fácil obtener la relación entre r y E:

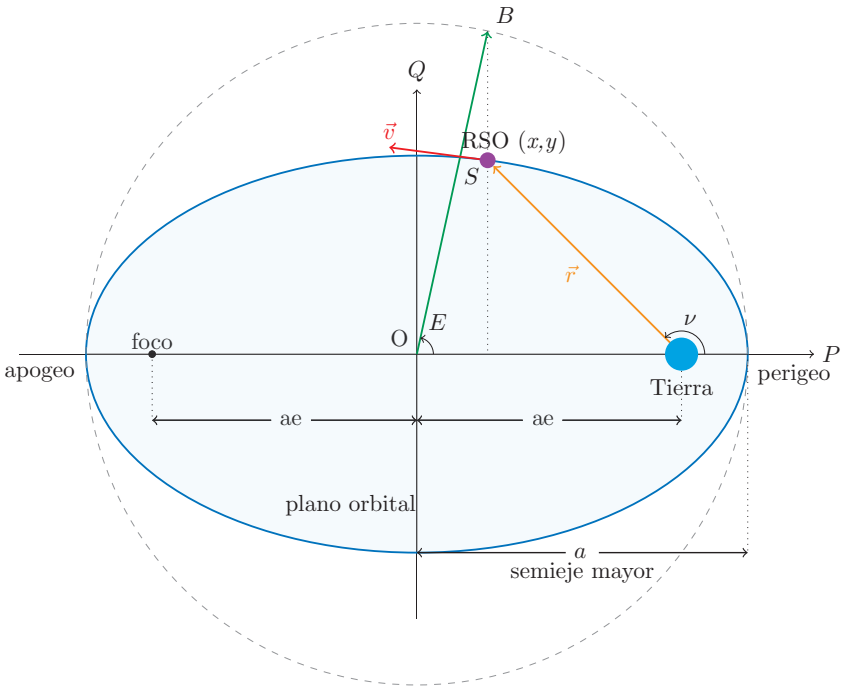


Figura 1-1.: Definición de la anomalía excéntrica E.

$$r = a(1 - e\cos E), \tag{1-13}$$

y la relación entre la velocidad aerolar h y E:

$$\begin{aligned} h &= x \cdot \dot{y} - y \cdot \dot{x} \\ &= a^2\sqrt{1-e^2}\dot{E}(1 - e\cos E). \end{aligned} \tag{1-14}$$

Por otro lado, teniendo en cuenta la ecuación (1-7) y que $A = e\mu$, h se puede expresar como:

$$h = \sqrt{\mu a(1-e^2)}. \tag{1-15}$$

Substituyendo (1-15) en (1-14), se obtiene

$$\dot{E}(1 - e\cos E) = n, \tag{1-16}$$

donde $n = \sqrt{\mu/a}$ y recibe el nombre de *movimiento medio*.

Finalmente, integrando con respecto al tiempo se obtiene la ecuación de Kepler

$$E - e\,\mathrm{sen}\,E = n(t - t_p), \tag{1-17}$$

donde t_p indica el tiempo de paso del satélite por el perigeo ($E = 0$). El término $M = n(t - t_p)$ se le denomina la *anomalía media*. Este valor varía 360 grados durante una revolución del

satélite, incrementándose uniformemente con el tiempo t. En lugar de describir la órbita a partir del tiempo en que el satélite pasa por el perigeo, se introduce el valor M_0 de la anomalía media en alguna época de referencia t_0.

$$M = M_0 + n(t - t_0). \tag{1-18}$$

El *periodo orbital* es el tiempo que le cuesta al satélite recorrer su órbita. Este periodo es inversamente proporcianal al movimiento medio n:

$$T = \frac{2\pi}{n} = 2\pi\sqrt{\frac{a^3}{\mu}}. \tag{1-19}$$

Esta relación representa esencialmente a la tercera Ley de Kepler (el cuadrado del periodo orbital de un planeta es proporcional al cubo del semieje mayor de su órbita).

1.1.3. La órbita en el espacio. Elementos orbitales

Como ya se ha comentado, los parámetros a y e describen el tamaño y la forma de la órbita del satélite, mientras que, M define su posición a lo largo de la órbita. Sin embargo, para posicionar el satélite en el espacio son necesarias tres cantidades angulares más (ver figuran **1-2**). Para ello consideraremos el sistema de coordenadas ecuatorial. Este sistema tiene su origen en el centro de la Tierra, el eje z apunta hacia el polo norte y el plano de referencia x, y lo forma el plano ecuatorial. Además, el eje x señala al *punto vernal*. Estos ángulos son:

- La inclinación i, que viene definido como el ángulo formado por la intersección del plano orbital y el plano ecuatorial terrestre. Una inclinación de más de 90° significa que el movimiento del satélite es retrógrado, es decir, está orbitando la Tierra en sentido contrario al de su rotación.

- La ascensión recta del nodo ascendente Ω, es el ángulo formado por la dirección del equinoccio vernal, eje x, y el nodo ascendente.

- El argumento del perigeo ω, es el ángulo formado por el nodo ascendente y la dirección del perigeo.

Los seis parámetros descritos a, e, i, Ω, ω y M junto con las anomalías E y ν, son conocidos como los *elementos orbitales*, los cuales determinan la posición y velocidad de un satélite en un tiempo dado t_0. Los elementos orbitales se pueden clasificar como variables rápidas o variables lentas. Las variables rápidas presentan grandes cambios durante una revolución del satélite. Entre estos elementos se encuentran las tres anomalías: la media, la verdadera y la excéntrica. Las variables lentas por su parte, presentan pequeños cambios durante la revolución del satélite. En estas últimas se encuentran, el semieje mayor, la excentricidad, la inclinación, la ascensión recta del nodo ascendente y el argumento del perigeo.

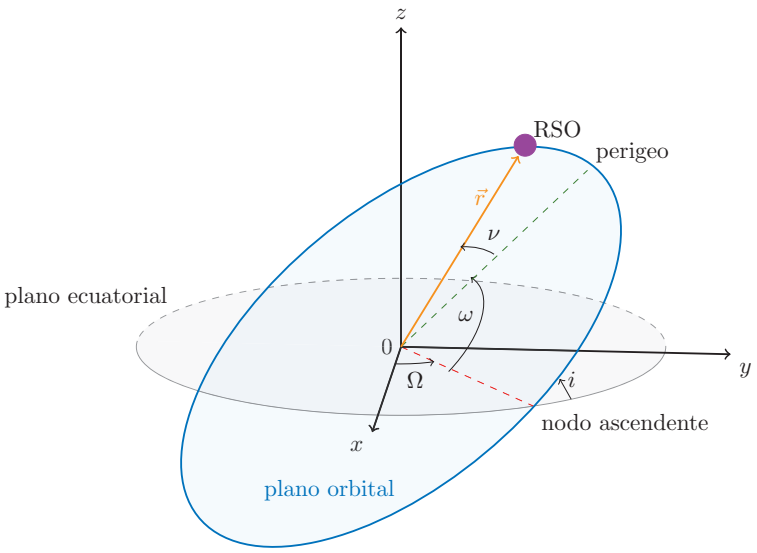


Figura 1-2.: Representación del plano orbital con respecto al sistema de coordenadas ecuatorial.

1.1.4. Tipos de órbitas

Las órbitas más comunes utilizadas por los satélites [29, 30] incluyen:

- Órbita Geoestacionaria (GEO).
- Órbita Terrestre Baja (LEO).
- Órbita Terrestre Media (MEO).
- Órbita Polar y Órbita Sincrónica Solar (SSO).
- Órbitas de Transferencia y Órbita de Transferencia Geoestacionaria (GTO).

La selección de la órbita se basa en función de los objetivos del satélite. A continuación se describe brevemente cada una de estas órbitas.

Órbitas Geoestacionarias

La órbita geoestacionaria, en inglés Geostationary Orbit (GEO), es aquella en la cual los satélites giran alrededor de la Tierra por encima del ecuador, con una velocidad que coincide con la rotación terrestre. Esto les confiere la apariencia de estar inmóviles sobre una posición fija. Su periodo orbital es de 23 horas, 56 minutos y 4 segundos. La órbita GEO resulta la más adecuada para los satélites de telecomunicaciones y de vigilancia meteorológica, ya que permiten monitorear constantemente una ubicación específica en la superficie terrestre. De modo que con tan solo tres satélites dispuestos en órbita GEO, se puede lograr una

cobertura casi global debido a su distancia a la Tierra, que ronda aproximadamente los 35 786 kilómetros.

Órbita Terrestre Baja

Una órbita terrestre baja, en inglés Low Earth Orbit (LOW), se encuentra a una altitud relativamente cercana a la Tierra, generalmente por debajo de los 1000 kilómetros, e incluso puede extenderse en casos particulares hasta los 160 kilómetros sobre la superficie terrestre. A diferencia de la órbita geoestacionaria, los satélites en órbitas bajas pueden seguir diversas trayectorias, lo que les proporciona más flexibilidad. La proximidad de la órbita LEO es beneficiosa para capturar imágenes de alta resolución desde los satélite. Además, esta es la órbita de la Estación Espacial Internacional (ISS por sus siglas en inglés), que facilita el transporte de astronautas y suministros desde la Tierra.

Los satélites en órbita LEO se desplazan a una velocidad de aproximadamente 7.8 kilómetros por segundo, con un periodo orbital de alrededor de 90 minutos. Debido a su rápido movimiento, no se utilizan satélites individuales en LEO para funciones como las telecomunicaciones. En su lugar, se crean constelaciones de satélites para garantizar una cobertura continua.

Órbita Terrestre Media

La órbita terrestre media, en inglés Medium Earth Orbit (MEO), abarca una amplia gama de órbitas entre las órbitas LEO y las GEO. Al igual que en las orbitas bajas, los satélites MEO se emplean para diversos fines ya que no siguen trayectorias específicas alrededor de la Tierra.

Por ejemplo, los satélites de navegación, como los que forman parte del sistema europeo Galileo, operan en órbitas MEO. El sistema Galileo desempeña un papel crucial en la navegación y las comunicaciones en toda Europa, facilitando tareas como el seguimiento de aviones y la provisión de direcciones precisas a dispositivos móviles. Esto se logra mediante el despliegue de varios satélites en una constelación, lo que asegura una amplia cobertura simultánea en extensas áreas del mundo.

Órbita Polar y Órbita Heliosincrónica

Teóricamente se define una orbita polar, como aquella que forma un ángulo de 90° grados con el plano ecuatorial terrestre. Sin embargo, esta definición estricta suele relajarse en la práctica para incluir cualquier orbita satelital que cruce por encima de ambos polos terrestres, con un ángulo de desviación de entre 20° y 30° grados. Las orbitas polares, son un tipo de órbitas LEO con una altitud comprendida entre 200 y 1000 km. Su periodo orbital es de aproximadamente 100 minutos. Debido a la inclinación del plano orbital, el planeta gira por debajo de una órbita polar, lo que le permite al satélite acceder prácticamente todos los puntos de la superficie terrestre. Esta característica hace que estas orbitas sean ideales para posicionar satélites para tareas de observación terrestre y/o investigación científica, por ejemplo, para la vigilancia de los cultivos, los bosques y la seguridad mundial.

La órbita heliosincrónica, en inglés Sun-Synchronous Orbit, (SSO), es un tipo particular de órbita polar. Esta órbita se caracteriza porque los satélites mantienen una posición constante con respecto al Sol, por lo que siempre pasan por el mismo lugar a la misma hora local sincronizándose para permanecer constantemente en el amanecer o atardecer, evitando así las zonas de sombra proyectadas por la Tierra. Los satélites en órbita heliosincrónica se sitúan a altitudes que oscilan entre 600 y 800 km. Este tipo de satélites suelen ser particularmente útiles para las comunicaciones, la cartografía terrestre y los estudios meteorológicos entre otras aplicaciones. Por ejemplo, la constelación de satélites Iridium [31] utiliza una órbita polar a una altitud de aproximadamente 780 km para ofrecer sus servicios de telecomunicaciones.

Órbitas de Transferencia y Órbita de Transferencia Geoestacionaria

Cuando un satélite se lanza al espacio, no siempre se coloca directamente en su órbita final. En algunos casos, el vehículo lanzador lo coloca en una órbita temporal. Posteriormente, utilizando sus propios motores, el satélite se traslada a una órbita de mayor altitud. Este proceso evita que el vehículo lanzador recorra toda la distancia necesaria, lo cual requeriría un esfuerzo considerable.

Estas órbitas temporales se conocen como órbitas de transferencia y se caracterizan por ser altamente elípticas. En una órbita de transferencia el perigeo puede estar a la altura de una órbita LEO, mientras que el apogeo, puede alcanzar altitudes similares a las de una órbita GEO.

Una de las órbitas de transferencia más comunes es la órbita de transferencia geoestacionaria (GTO por sus siglas en inglés, Geostationary Transfer Orbit), empleada como una etapa intermedia para posicionar los satélites en la órbita GEO.

1.1.5. Puntos de Lagrange

Los puntos de Lagrange son lugares específicos en el espacio en los que las influencias gravitatorias de la Tierra y el Sol interactúan de manera que las naves espaciales pueden mantener órbitas estables. Estos puntos están situados a más de un millón de kilómetros. A diferencia de las órbitas convencionales, las naves espaciales situadas en los puntos de Lagrange no orbitan directamente alrededor de la Tierra. En lugar de ello, se colocan en posiciones específicas, aprovechando las fuerzas gravitatorias combinadas de la Tierra y el Sol.

Las naves espaciales situadas en los puntos de Lagrange pueden permanecer relativamente cerca de la Tierra con un consumo mínimo de energía, lo que permite mantener posiciones estables y fijas. Esta estrategia resulta ventajosa en comparación con el lanzamiento de naves hacia regiones lejanas del espacio, donde naturalmente caerían en órbita alrededor del Sol. En tales casos, a medida que la nave se aleja, la comunicación con la Tierra se vuelve más complicada.

El uso estratégico de los puntos de Lagrange ofrece una solución práctica al permitir que las naves espaciales permanezcan cerca de la Tierra, evitando las complicaciones asociadas con las órbitas solares lejanas. Dos de los puntos de Lagrange más utilizados son L1 y L2, ubicados a una distancia de 1,5 millones de kilómetros de la Tierra, cuatro veces la distancia entre

la Tierra y la Luna. Estos puntos son empleadas por observatorios y telescopios espaciales, ya que su distancia a la Tierra evita que la luz visible y la radiación infrarroja emitida por nuestro planeta interfieran con sus mediciones.

1.2. Perturbaciones

Al comparan las efemérides precisas de cualquier RSO con las predichas por el modelo de Kepler, se observa una discrepancia entre ambas. Esta diferencia entre la trayectoria real y las predicciones teóricas se debe a la existencia de perturbaciones. Una *perturbación*, se define como una desviación del movimiento normal o esperado que se produce como resultado de las fuerzas externas que no son consideradas en el modelo de Kepler. Las perturbaciones pueden ser de naturaleza gravitacional, como son las ocasionadas por la no esfericidad de la Tierra o por la presencia de terceros cuerpos, como pueden ser el Sol, la Luna u otros planetas, entre otras. También pueden ser ocasionadas por factores impredecibles como la dirección y fuerza del viento solar o la presión de radiación solar. Mientras que las perturbaciones impredecibles se tratan de manera estocástica, las gravitacionales se abordan mediante modelos deterministas [32, 33, 34, 35, 36].

Las perturbaciones no siempre son fuerzas pequeñas; de hecho, pueden ser tan significativas como la fuerza que define el modelo de Kepler. Por ejemplo, en el caso de los satélites, ignorar las irregularidades producidas por el campo gravitatorio terrestre podría ocasionar una pérdida en la precisión de la predicción de la posición del satélite a largo plazo. El número total de perturbaciones a considerar puede variar dependiendo de las características tanto del satélite como de la trayectoria que describe. La figura **1-3** muestra el efecto que producen algunas de las perturbaciones sobre un satélite en función de su distancia al centro de la Tierra.

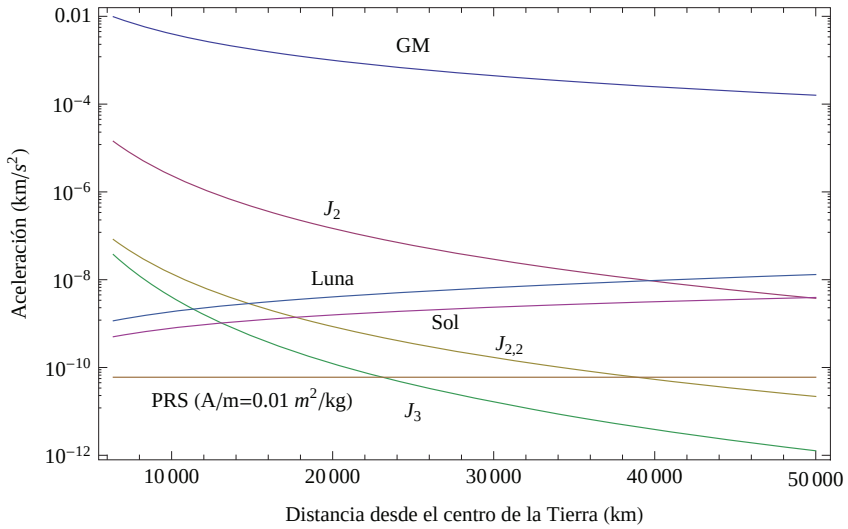


Figura 1-3.: Orden de magnitud, en escala logarítmica, de algunas fuerzas de perturbación.

Sin embargo, al mejorar la precisión del modelo de Kepler incluyendo nuevas perturbaciones tiene como consecuencia que los elementos orbitales, considerados constantes en el modelo de Kepler, pasan a ser variables. Los efectos producidos por las perturbaciones se clasifican en dos tipos: *seculares* y *periódicos* (de corto y largo periodo), dependiendo del efecto que produzcan sobre los elementos orbitales. Los distintos tipos de variaciones se pueden apreciar en la figura **1-4**.

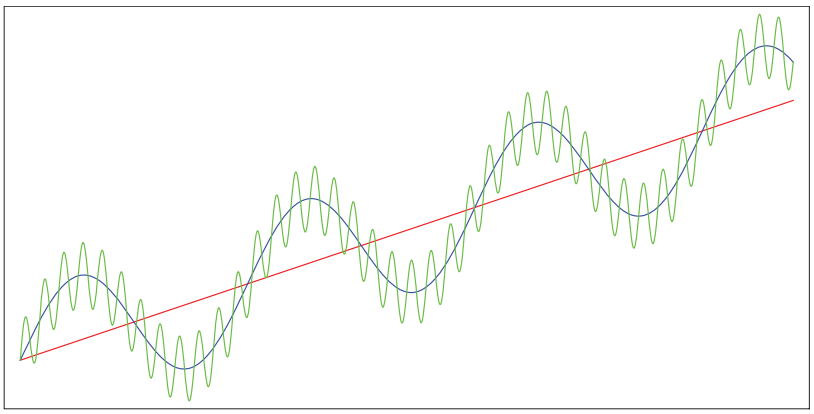


Figura 1-4.: La línea verde muestra la evolución real de un elemento orbital, incluyendo todos los términos. La línea azul representa la evolución de los términos seculares y de largo periodo, mientras que la línea roja muestra únicamente los términos seculares.

Los efectos seculares generan variaciones lineales en los elementos orbitales con respecto al tiempo, a veces exhibiendo comportamientos que incluyen potencias del tiempo. Por otro lado, los efectos periódicos introducen un comportamiento oscilatorio en los elementos orbitales. Estos efectos se clasifican en corto y largo plazo según el tiempo requerido para completar una órbita del satélite. Los efectos de corto plazo se caracterizan por repetirse en un tiempo igual o menor al periodo orbital, mientras que los de largo plazo oscilan en ciclos mayores, generalmente entre uno y dos órdenes de magnitud del periodo orbital [13].

Para describir con mayor precisión la trayectoria de un satélite, se introduce un modelo de fuerzas más complejo en la ecuación (1-1):

$$\ddot{\boldsymbol{r}} = \mathbf{a}_{\mathcal{V}} + \mathbf{a}_{3rd} + \mathbf{a}_{drag} + \mathbf{a}_{prs} + \mathbf{a}_{O}, \tag{1-20}$$

donde $\mathbf{a}_{\mathcal{V}}$ representa la aceleración producida por el campo gravitatorio terrestre, en el cual se incluye el término $-(\mu/r^3)\boldsymbol{r}$ que proviene del problema de Kepler. Las aceleraciones $\mathbf{a}_{3rd}$, $\mathbf{a}_{drag}$ y $\mathbf{a}_{prs}$ corresponden, respectivamente, a los efectos producidos por el tercer cuerpo, el rozamiento atmosférico y la presión de radiación solar. El último término de la ecuación (1-20) recoge las aceleraciones producidas por pequeñas perturbaciones, del orden de 10^{-15} y 10^{-12} Km/s^2, y se descompone en

$$\mathbf{a}_{O} = \mathbf{a}_{T} + \mathbf{a}_{rel} + \mathbf{a}_{RI} + \mathbf{a}_{A} + \mathbf{a}_{Ey} + \mathbf{a}_{Sy}, \tag{1-21}$$

donde $\mathbf{a}_T$ representa la aceleración producida por las fuerzas de marea, $\mathbf{a}_{rel}$ por los efectos relativistas, $\mathbf{a}_{RI}$ por la radiación infrarroja, $\mathbf{a}_A$ por el albedo de la Tierra y $\mathbf{a}_{Ey}$ y $\mathbf{a}_{Sy}$ por las fuerzas de Yarkovsky producidas por la Tierra y el Sol, respectivamente. Las expresiones explícitas de estas aceleraciones pueden verse en [26, 37, 13].
A continuación, se describen brevemente las principales fuerzas que perturban el movimiento de un satélite artificial, así como los efectos que producen sobre los elementos orbitales. Estas perturbaciones incluyen el campo de gravedad de la Tierra, los efectos producidos por el Sol y la Luna, el rozamiento atmosférico y la presión de radiación solar.

1.2.1. Campo de gravedad de la Tierra

El campo de gravedad ejercido por la Tierra sobre un objeto que se encuentra en un punto P a una distancia r de su centro de masas se expresa en coordenadas esféricas (r, λ, β) como:

$$\mathcal{V} = -\frac{\mu}{r}\sum_{n\geq 0}\left(\frac{\alpha}{r}\right)^n \sum_{0\leq m\leq n}(C_{nm}\cos m\lambda + S_{nm}\operatorname{sen} m\lambda)P_n^m(\operatorname{sen}\beta), \tag{1-22}$$

donde α representa el radio ecuatorial de la Tierra. Los parámetros C_{nm} y S_{nm} son constantes adimensionales, con $C_{00} = 1$ y C_{10}, C_{11}, S_{11} iguales a cero, llamados coeficientes armónicos. Estos coeficientes describen la dependencia del campo con la distribución interna de masas de la Tierra.

Las funciones P_n^m son los polinomios asociados de Legendre de grado n y orden m, definidos como:

$$P_n^m(x) = \frac{(1-x^2)^{m/2}}{2^n n!}\frac{d^{n+m}}{dx^{n+m}}(x^2-1)^n. \tag{1-23}$$

Los índices m y n definen tres tipos de coeficientes armónicos, que proporcionan una idea de la forma y distribución de masas de la Tierra. Los coeficientes armónicos C_{nm} con m=0, notados usualmente como $J_n = -C_{n0}$, son llamados coeficientes zonales y describen un campo simétrico alrededor del eje de rotación, formando una especie de bandas de latitud. Los coeficientes armónicos con $m = n$, son conocidos como los Armónicos sectoriales y representan usos en longitud sobre la Tierra. Por su parte, a los coeficientes armónicos donde $m < n$ se les denomina Armónicos teserales, ya que dividen la superficie de la esfera, en sectores similares a un tablero de ajedrez, ver figura **1-5**.

La aceleración producida por el campo de gravedad de la Tierra es

$$\mathbf{a}_{\mathcal{V}} = \nabla_{\boldsymbol{r}}\mathcal{V}. \tag{1-24}$$

Es importante destacar, que $-(\mu/r^3)$ es el término preponderante en el potencial gravitacional terrestre, indicando virtualmente la concentración de toda la masa en el centro de la Tierra. Los términos restantes exhiben magnitudes decrecientes. De todos los coeficientes, $C_{20} = -J_2$, que caracteriza la asimetría del campo gravitacional terrestre atribuible al achatamiento polar, destaca por su magnitud, situándose en el orden de 10^{-3}, mientras que los demás alcanzan valores del orden de 10^{-6} o inferiores. Los armónicos zonales pares generan

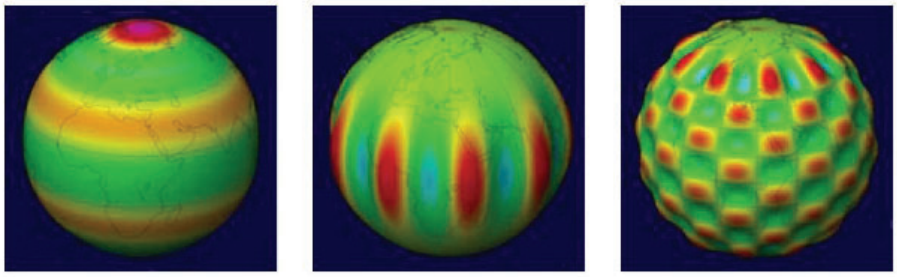


Figura 1-5.: Tipos de armónicos esféricos: zonal con P_n^0 (izquierda); sectorial con P_n^n y $n \neq 0$ (centro); teseral con P_n^m y $n \neq m \neq 0$ (derecha).

efectos seculares, mientras que los impares provocan únicamente variaciones periódicas. Por su parte, los armónicos teserales y sectoriales provocan únicamente variaciones periódicas.

1.2.2. Atracción del tercer cuerpo (Sol y Luna)

El efecto producido por el Sol y la Luna es poco significativo en los satélites que describen orbitas cercanas a la Tierra. No obstante, en órbitas más alejadas de la Tierra donde la influencia del frenado atmosférico es menor, el efecto del tercer cuerpo puede ser igual o incluso más importante que el achatamiento terrestre. Este tipo de perturbaciones se debe a la interacción gravitacional, por lo que se trata de una aceleración provocada por una fuerza conservativa. La aceleración producida por estos dos cuerpos se puede describir mediante la ecuación [13]:

$$\mathbf{a}_{3rd} = -\sum_{j=1}^{2} \mu_{3rd}^{j} \left[\frac{\mathbf{r}_{\oplus 3b}^{j}}{\|\mathbf{r}_{\oplus 3b}^{j}\|^3} - \frac{\mathbf{r}_{sat3b}^{j}}{\|\mathbf{r}_{sat3b}^{j}\|^3} \right], \tag{1-25}$$

donde μ_{3rd} representa la constante de gravedad del tercer cuerpo, $\mathbf{r}_{\oplus 3b}$ es la distancia entre la Tierra y el tercer cuerpo, y $\mathbf{r}_{sat3b}$ la distancia entre el satélite y el tercer cuerpo. Para $j = 1$ el tercer cuerpo es el Sol y para $j = 2$ la Luna.

Los efectos más significativos de estas perturbaciones tienen lugar en la excentricidad e inclinación de la orbita. Sin embargo, estas perturbaciones no afectan su energía, y por lo tanto, no producen ninguna variación secular sobre el semieje mayor.

1.2.3. Rozamiento atmosférico

El rozamiento atmosférico se debe a la energía cinética transmitida por el choque de las partículas atmosféricas con el satélite. Al ser una perturbación dependiente de la velocidad, es una fuerza no conservativa que afecta principalmente el semieje mayor de la órbita y su excentricidad. Esta fuerza actúa en dirección opuesta a la velocidad del satélite relativa a la atmósfera, por lo que reduce su tiempo de vida y ocasiona efectos periódicos sobre los demás elementos orbitales.

La aceleración producida por el rozamiento atmosférico puede expresarse como [13]:

$$\mathbf{a}_{drag} = -\frac{1}{2} C_D \left(\frac{A}{m}\right) \rho \mathrm{v}_{rel}^2 \frac{\mathbf{v}_{rel}}{\|\mathbf{v}_{rel}\|}, \tag{1-26}$$

donde C_D, es un parámetro adimensional llamado coeficiente de rozamiento atmosférico. Este coeficiente describe la interacción de la atmósfera con la superficie del satélite. Este coeficiente generalmente se aproxima a 2.2; no obstante, dependiendo de la forma del satélite, puede variar entre 1.5 y 3.0. La variable ρ representa la densidad de la atmósfera con respecto a la altitud a la que se encuentra el satélite. Determinar este parámetro resulta complejo debido a la influencia de varios factores, como la estructura molecular de la atmósfera, la incidencia del flujo solar y los campos geomagnéticos del Sol y la Tierra. A menudo, se emplea un modelo que asume una disminución exponencial de la densidad en relación con la altura:

$$\rho = \rho_0 e^{-\frac{h-h_0}{H}}, \tag{1-27}$$

donde ρ_0 representa el valor de la densidad de referencia a la altitud h_0, mientras que H es un parámetro denominado factor de escala y h denota la altitud local del satélite.

En la ecuación 1-26, el término A/m corresponde al cociente entre el área y la masa del satélite, la cual no siempre puede considerarse constante. El factor $B = C_D A/m$ se conoce como coeficiente balístico. Por último, $\mathbf{v}_{rel}$ representa la velocidad relativa entre el satélite y la atmósfera, la cual varía en función de la altura con respecto a la Tierra. Esta velocidad se determina mediante la siguiente expresión:

$$\mathbf{v}_{rel} = \frac{d\boldsymbol{r}}{dt} - (\boldsymbol{\omega}_{\oplus} \times \boldsymbol{r}), \tag{1-28}$$

donde $\boldsymbol{\omega}_{\oplus}$ representa la velocidad angular de rotación de la Tierra.

Si la distancia en el perigeo es inferior a los 200 km, esta perturbación puede provocar que el satélite se desvíe de su órbita en tan solo unos pocos días. En cambio, si la órbita es altamente elíptica, la influencia de la fuerza de rozamiento afecta principalmente al movimiento del satélite durante su paso por el perigeo, mientras que en el resto de la órbita, donde las altitudes son mayores, no se observa una perturbación significativa. Aunque en este caso la distancia al perigeo no se reduce; sin embargo, la velocidad en la dirección tangente a la órbita experimenta una ligera disminución. Esto tiende a igualar la velocidad máxima del satélite durante el paso por el perigeo, con la velocidad mínima durante el paso por el apogeo. Como resultado, se produce la circularización de la órbita satelital, es decir, una reducción de la excentricidad orbital.

Por último, es importante destacar que, en términos generales, los elementos orbitales experimentan perturbaciones de corto periodo, especialmente i, Ω y ω

1.2.4. Presión de radiación solar

La presión de radiación solar se origina por la transferencia de energía cinética a través de las colisiones de los fotones solares con la superficie del satélite, mayormente reflejados en la dirección del vector $\mathbf{r}_{sat\odot}$, que conecta el Sol con el satélite. Al igual que el rozamiento

atmosférico, es una fuerza no conservativa que afecta la excentricidad orbital, especialmente de los satélites provistos con grandes paneles solares, como es el caso de muchos satélites geoestacionarios.

La aceleración que un satélite experimenta por la presión de radiación solar, puede describirse mediante el siguiente modelo simplificado:

$$\mathbf{a}_{prs} = -\frac{p_{sr} C_r A}{m} \frac{\mathbf{r}_{sat\odot}}{\|\mathbf{r}_{sat\odot}\|}, \tag{1-29}$$

donde p_{sr} es la constante de presión de radiación solar, C_r es el coeficiente de reflectividad, cuyo valor varía entre 0 y 2, y A y m son respectivamente la superficie y la masa del satélite.

El efecto de esta perturbación es complicado de modelizar ya que depende de la determinación precisa de los ciclos solares y sus variaciones. Además, se anula en las zonas donde se proyecta la sombra de la Tierra. El efecto producido por esta perturbación, está relacionado con el movimiento de la Tierra alrededor del Sol, por lo que se clasifica como una perturbación de largo periodo con un periodo de un año. En ocasiones la altura del perigeo se puede ver afectada, y a menor altura mayor rozamiento atmosférico, por lo que se afecta el tiempo de vida del satélite. El efecto producido por la presión de radiación solar puede superar al del rozamiento atmosférico alrededor de los 500 km de altura, esto dependiendo de la relación que exista entre el área y la masa del satélite.

1.3. Integración de la órbita

Las técnicas clásicas empleadas para la integración de las ecuaciones del movimiento (1-20) pueden agruparse en dos grandes categorías: teorías especiales de perturbaciones y teorías generales de perturbaciones. Las primeras se caracterizan por integrar numéricamente las ecuaciones del movimiento. En este caso, es fácil añadir una nueva perturbación al sistema de ecuaciones diferenciales, para ello se expresa la nueva aceleración en función del tiempo y del estado del objeto, es decir a través de su posición y de su velocidad. Sin embargo, uno de los factores más importantes a la hora de utilizar un integrador numérico es la selección del tamaño del paso de integración. Este factor es el que determina la precisión de la solución y, posteriormente, su eficiencia computacional. Las teorías generales directamente proporcionan una solución analítica aproximada de las ecuaciones del movimiento. Estas teorías suelen ser mucho más rápidas que sus homólogas numéricas, pero normalmente son más difíciles de formular, más difíciles de programar y no son tan precisas. Por último, existe una tercera técnica a considerar, las teorías semi-analíticas, las cuales combinan las dos técnicas anteriores de modo que se evitan los inconvenientes que presentan cada una de ellas por separado.

A partir de aquí se define el concepto de *propagador orbital* como una implementación en un lenguaje de programación de una de las soluciones anteriores. Por lo tanto, cada propagador orbital dependerá, primero, del modelo de fuerzas considerado en las ecuaciones del movimiento, así como del método numérico o la teoría de perturbaciones empleada en la obtención de la solución, segundo, de las especificaciones técnicas de la implementación, como pueden ser el lenguaje de programación o el paradigma, entre otros.

En estos momentos, el creciente interés por el uso del espacio, tanto desde el ámbito civil como del militar, se traduce en un incremento continuo de los desechos espaciales. Este incremento ha hecho que se compliquen las operaciones relacionadas con la seguridad y vigilancia del espacio, y en consecuencia con el aumento de los requisitos que deben cumplir los sistemas de propagación y determinación orbital, en particular, el aumento de su precisión sin un excesivo incremento de su coste computacional. La mejora de estos sistemas sólo se puede alcanzar o bien ampliando el modelo de fuerzas o bien seleccionando un integrador numérico más preciso, en el caso de las teorías especiales, o proporcionando ordenes más altos de las aproximaciones analíticas, en el caso de las teorías generales. En cualquier caso, estas mejoras implican la modificación del código del correspondiente propagador orbital. En 2007, San-Juan propuso una cuarta técnica que denominó *metodología híbrida*. Esta nueva técnica combina cualquiera de las tres anteriores con métodos estadísticos o de Inteligencia Artificial para mejorar el modelo de fuerzas o la técnica de integración. Como consecuencia, al correspondiente propagador orbital se le añade un módulo externo que proporciona una estimación de los efectos de las fuerzas no consideradas, las incertidumbres del modelo o la falta de precisión de las técnicas de integración. El propagador resultante se denomina *propagador orbital híbrido*, y es considerada una técnica no invasiva.

Acontinuación se introduccen brevemente las teorías y los propagadores orbitales empleados en el desarrollo de este trabajo.

1.3.1. Técnicas especiales de perturbaciones: método de Cowell

Como ya se ha visto, las ecuaciones del movimiento de un satélite se pueden expresar como un sistema de tres ecuaciones diferenciales de segundo orden:

$$\ddot{\boldsymbol{r}} + \frac{\mu}{r^3}\boldsymbol{r} = \boldsymbol{a}_{\mathcal{P}}, \tag{1-30}$$

o de forma equivalente, a través de un sistema de seis ecuaciones diferenciales de primer orden:

$$\begin{aligned} \dot{\boldsymbol{r}} &= \boldsymbol{v}, \\ \dot{\boldsymbol{v}} &= -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{a}_{\mathcal{P}}, \end{aligned} \tag{1-31}$$

donde $\boldsymbol{a}_{\mathcal{P}}$ representa a todas las aceleraciones que perturban la trayectoria del satélite. En el caso de que $\boldsymbol{a}_{\mathcal{P}} = \boldsymbol{0}$ se recupera el problema de Kepler y se puede resolver analíticamente. Sin embargo, ésto no es cierto en general cuando $\boldsymbol{a}_{\mathcal{P}} \neq \boldsymbol{0}$.

La integración numérica de las ecuaciones del movimiento, ya sea (1-30) o (1-31), recibe el nombre de *método de Cowell*. Este método proporciona la solución más precisa de todas las posibles, ya que no requiere simplificar ni aproximar las ecuaciones del movimiento para su integración, como lo hacen, por ejemplo, las teorías generales de perturbaciones o los métodos semi-analíticos. Sin embargo, este método no es el más eficiente desde el punto de vista computacional, ya que depende del tamaño del paso de integración. Si las perturbaciones consideradas son del mismo orden o superior a $-\mu/r^3\boldsymbol{r}$, el tamaño del paso es razonable; sin

embargo, este disminuye si la perturbación es menor, lo que hace que el método de Cowell se vuelva ineficiente.

En esta memoria se va a utilizar el propagador numérico de alta precisión AIDA (Accurate Integrator for Debris Analysis) [38] para simular las pseudo-observaciones utilizadas en este trabajo. AIDA es un propagador numérico especialmente adaptado para el análisis de los residuos espaciales utilizando las técnicas que proporciona el álgebra diferencial de Taylor (DA) [39]. Las perturbaciones incluidas en AIDA son el campo de gravedad terrestre, las perturbaciones producidas por el Sol y la Luna, el rozamiento atmosférico (NRLMSISE-00) y la presión de la radiación solar. El integrador numérico que utiliza es el método Runge-Kutta de Dormand-Prince [40].

1.3.2. Técnicas especiales de perturbaciones: método de Encke

El método de Encke fue propuesto por el astónomo aleman Johann Franz Encke a principios del siglo XIX para determinar con precisión las órbitas de cometas. El método considera que si en un instante determinado t_0 el término de la derecha de la ecuación (1-30) es nulo, es decir, que en t_0 ninguna de las fuerzas consideradas perturba el movimiento del objeto, este describirá una trayectoria elíptica que puede ser calculada utilizando las fórmulas del problema de Kepler a partir de la posición y velocidad del objeto en t_0, $(\boldsymbol{r}_0, \boldsymbol{v}_0)$. Esta elipse instantánea recibe el nombre de osculante o de referencia. Por supuesto, el movimiento real del objeto nunca tiene lugar a lo largo de la órbita osculante; sin embargo, si las fuerzas perturbadoras son pequeñas, comparadas con el término $-(\mu/r^3)\boldsymbol{r}$, entonces, en periodos cortos de tiempo, la posición real del objeto en la órbita real diferirá de la posición asociada en la órbita osculante en una cantidad pequeña. El método de Encke se basa en el análisis de esta diferencia.

En términos matemáticos, en el método de Encke se consideran las trayectorias de dos objetos. El primero describe la órbita osculatriz, mientras que el segundo la trayectoria real. Por otro lado, sus posiciones quedan determinadas por los vectores $\boldsymbol{r}_{osc}$ y $\boldsymbol{r}$, respectivamente, y su evolución temporal por las ecuaciones (1-1) y (1-30). A continuación, se define la distancia entre los dos objetos como:

$$\boldsymbol{\rho} = \boldsymbol{r} - \boldsymbol{r}_{osc}. \tag{1-32}$$

y su evolución temporal como:

$$\ddot{\boldsymbol{\rho}} = \ddot{\boldsymbol{r}} - \ddot{\boldsymbol{r}}_{osc}. \tag{1-33}$$

Entonces, sustituyendo (1-1) y (1-30) en (1-33) se obtiene que:

$$\ddot{\boldsymbol{\rho}} = -\frac{\mu}{r_{osc}^3}\left[\left(\frac{r_{osc}^3}{r^3} - 1\right)\boldsymbol{r} - \boldsymbol{\rho}\right] + \boldsymbol{a_p}, \tag{1-34}$$

donde $r_{osc} = |\boldsymbol{r}_{osc}|$ y $r = |\boldsymbol{r}|$. Es importante destacar que si la perturbación es muy pequeña pueden aparecer inestabilidades numéricas debido a que el término r_{osc}^3/r^3 tiende a uno. En [28] se describe una técnica para evitar este problema.

La eficiencia computacional del método de Encke se debe a que $\boldsymbol{\rho}$ cambia más lentamente que $\boldsymbol{r}$, por lo que es posible utilizar pasos más largos en la integración que en el método de Cowell. No obstante, a medida que avanza el tiempo, las diferencias entre las órbitas real y osculante se van haciendo más grandes, especialmente en el caso de perturbaciones seculares. Por esta razón, es necesario realizar una rectificación de la órbita de referencia para que coincida con la órbita real.

1.3.3. SGP4

SGP4 es un propagador orbital utilizado para calcular el vector de estado de un RSO (posición y velocidad) en un sistema de coordenadas inerciales centrado en la Tierra [41]. Desarrollado por el Comando de Defensa Aérea de América del Norte (NORAD), SGP4 se basa en las teorías analíticas de Kozai y Brouwer [42, 43, 14]. En sus inicios, SGP4 consideraba solo las perturbaciones causadas por los armónicos zonales hasta J_4 y el rozamiento atmosférico, relevantes principalmente en órbitas LEO. Con el avance y el diseño de misiones más allá de las órbitas cercanas a la Tierra, se ha mejorado el modelo de perturbaciones de este propagador. En la actualidad, SGP4 incorpora efectos debidos a la forma de la Tierra, el rozamiento atmosférico y los efectos gravitatorios del Sol y la Luna. Es importante mencionar que SGP4, según el tipo de órbita, selecciona y aplica solo las perturbaciones más influyentes sobre la trayectoria del RSO. Para una descripción detallada del modelo matemático implementado en el propagador SGP4, se pueden consultar las referencias [44, 45, 46].

SGP4 solo puede utilizar como datos de entrada un archivo conocido como Two Line Elements (TLE), que contiene la información estandarizada de los elementos orbitales medios y un intervalo de tiempo especificado en minutos. A partir de esta información, el algoritmo realiza una comprobación del periodo orbital y la excentricidad para determinar qué perturbaciones debe incluir. El formato empleado en los TLE fue especificado por NORAD y continúa siendo utilizado en la actualidad. En la figura **1-6** se muestra, como ejemplo, el TLE de la Estación Espacial Internacional, junto con una descripción de cada elemento.

Figura 1-6.: Ejemplo de un TLE de la Estación Espacial Internacional (ISS).

2. Conceptos básicos de IA

En este capítulo se realiza una breve introducción de algunos de los conceptos básicos de la Inteligencia Artificial, haciendo especial hincapié en los modelos de redes neuronales. A continuación, se describe el proceso que se sigue para entrenar una red neuronal. Para finalizar, se muestra un ejemplo de predicción de series temporales univariadas utilizando modelos de redes neuronales.

2.1. Inteligencia Artificial

Según el grupo de expertos en Inteligencia Artificial de la Comisión Europea AI HLEG (por las siglas en inglés de High-level Expert Group on Artificial Intelligence) [47], la Inteligencia Artificial (IA), hace referencia a los "Sistemas que muestran un comportamiento inteligente analizando su entorno y emprendiendo acciones con cierto grado de autonomía, para alcanzar objetivos específicos". Aunque esta definición goza de buena aceptación por parte de la comunidad científica, no es la única. No obstante, para los objetivos de este trabajo resulta apropiada por dos razones. Primero, permite distinguir la IA del simple uso de algoritmos y de la tecnología digital en general. Segundo, incluye el aprendizaje profundo, también conocido como *Deep Learning* en inglés, y un campo dentro del aprendizaje automático denominado *Machine Learning* en inglés, aspectos cruciales en el desarrollo de esta investigación.

En sus inicios, la IA se concebía como el desarrollo de sistemas capaces de resolver tareas complejas para el ser humano, como diseñar una estrategia ganadora en una partida de ajedrez [48]. Sin embargo, a medida que la IA avanzaba, se observó que tareas como esta podían resolverse mediante un conjunto de reglas matemáticas formales [49]. Desde esta perspectiva, se reducía la IA a la simple ejecución de algoritmos por un ordenador, lo cual no podía considerarse como un comportamiento inteligente [50]. Debido a esto, la IA comenzó a relacionarse con el desarrollo de sistemas capaces de resolver problemas que se consideran intuitivos para los seres humanos. Sin embargo, problemas como el reconocimiento de objetos en imágenes o palabras en grabaciones de voz, resulta difícil de reducir a un conjunto de reglas formales.

2.1.1. Aprendizaje automático

Las dificultades encontradas para describir las tareas cotidianas en términos de reglas formales pusieron de manifiesto la necesidad de dotar a los ordenadores con la capacidad de obtener su propio conocimiento y, a su vez, reducir la intervención humana en la solución de las tareas planteadas. Este conocimiento se adquiere extrayendo patrones y reglas a partir de los datos originales del problema que se quiere resolver. A esta capacidad se le denomina *aprendizaje automático* o *Machine Learning* (ML) [51].

La eficiencia de los algoritmos de ML depende principalmente de cómo se representen los datos. Una *representación* hace referencia a la forma en que se muestran los datos. Por ejemplo, para determinar el estado de un cultivo se pueden utilizar los datos provenientes de un análisis químico de las hojas, lo que generará una tabla con datos numéricos. También se puede determinar su estado mediante una cámara multiespectral, en cuyo caso, se contarán con datos en forma de imágenes. A cada pieza de información incluida en una representación se le denomina *característica*. En función del problema y de los datos disponibles, se distinguen tres tipos de ML: *aprendizaje supervisado*, *aprendizaje no supervisado* y *aprendizaje por refuerzo*. En la sección 2.1.4 se introduce brevemente el aprendizaje supervisado, mientras que para los aprendizajes no supervisados y por refuerzo se puede consultar [52, 53, 54].

Las técnicas de ML son aplicables siempre que se disponga de las características adecuadas [55]. Por ejemplo, las medidas craneales son bastante útiles para determinar el sexo a partir de los restos óseos de un ser humano [56]. Los problemas surgen cuando no es posible determinar con precisión las características que se deben extraer de los datos para, posteriormente, suministrarlas al algoritmo de aprendizaje. Esto se debe, en ocasiones, a que la representación de los datos de donde se pretenden extraer las características no es la adecuada. Por ejemplo, si se desea que un programa de ordenador aprenda a reconocer automóviles a partir de imágenes, una característica a buscar pueden ser las ruedas. Esta tarea se reduce a reconocer círculos con diferentes orientaciones en las imágenes. Sin embargo, como la imagen es leída por el ordenador como una tabla de valores donde cada número está determinado por el color de cada píxel, factores como el reflejo de la luz sobre la rueda o las sombras dificultan la tarea, ya que modifica el color de cada píxel.

2.1.2. Aprendizaje de representaciones

El problema de la extracción adecuada de características se soluciona permitiendo que el ordenador no solo aprenda a relacionar las representaciones de los datos con las salidas o respuestas, sino que también aprenda a seleccionar la mejor representación en sí misma. Este enfoque dentro del ML, conocido como *aprendizaje de representaciones*, o por su denominación en inglés *Representation Learning* (RL) [57], reduce aún más la intervención humana en el desarrollo de sistemas inteligentes. El ejemplo más reconocido de RL son los *autocodificadores* o *autoencoders* [58]. Un autoencoder es la combinación de una función codificadora que transforma los datos de entrada en una representación diferente, junto con una función decodificadora que devuelve los datos a su formato original. Los autoencoders usualmente son entrenados para convertir datos de alta dimensión en datos de baja dimensión preservando la máxima información posible [59, 60, 61]. El objetivo es que la representación obtenida por la función codificadora presente buenas propiedades, como una extracción más sencilla de características relevantes para el algoritmo de aprendizaje. Diferentes tipos de autoencoders generan diferentes tipos de representaciones de los datos [62].

2.1.3. Aprendizaje profundo

Cuando se diseñan algoritmos de IA para extraer características, el objetivo es separar, por un lado, los factores de variación que explican los datos observados y, por el otro, descartar los que no son importantes [55]. Estos factores afectan a cada unidad de información que se puede observar; si los factores de variación cambian, los datos observados también cambian. Por ejemplo, factores como la edad, el sexo y el acento de un hablante afectan el registro de los datos que se obtienen de una grabación.

En este punto es donde aparece el *aprendizaje profundo*, también conocido como *Deep Learning* (DL) [51]. El DL soluciona el problema de los factores de variación al introducir representaciones que se expresan en términos de otras más simples [48]. Esto permite al ordenador aprender a identificar conceptos complejos a partir de conceptos más simples. Por ejemplo, para determinar si una imagen contiene automóviles, la técnica del aprendizaje profundo no busca desde el principio elementos circulares como las ruedas. El proceso comienza buscando conceptos simples, como pueden ser las líneas rectas. Esto le permite construir un concepto más complejo, como una esquina o un contorno, lo que a su vez le permite desarrollar conceptos tales como una rueda o una puerta, y finalmente construye el concepto de lo que puede ser un automóvil.

El ejemplo más representativo del DL son las redes neuronales de propagación hacia adelante, también conocidas como *redes feed-forward*. Este es el tipo de redes comúnmente utilizado en el aprendizaje supervisado y es la herramienta empleada en el desarrollo de este trabajo, por lo que requiere una descripción más detallada.

2.1.4. Aprendizaje supervisado

En el campo de la IA, el aprendizaje supervisado se entiende como un proceso mediante el cual un ordenador, tras ser entrenado con datos, adquiere suficiente experiencia para resolver un problema específico [63]. Este tipo de aprendizaje se relaciona principalmente con problemas de clasificación y regresión

Problemas de clasificación

En los problemas de clasificación, el objetivo es construir un modelo clasificador que, a partir de un conjunto de datos de entrada con muestras agrupadas en clases, aprenda a diferenciar las características que definen a cada una de esas clases. Una vez obtenido el modelo, se puede emplear para inferir la clase asociada a nuevas muestras cuya clasificación sea desconocida.

Formalmente, un modelo de clasificación se puede definir de la siguiente manera: dado un conjunto de entrada que contiene K clases etiquetadas como y_k, se denota a $Y = \{y_k | 1 \leq k \leq K\}$ como el conjunto de clases del problema de clasificación. Un modelo clasificador es una función, $f : X \to Y$ definida sobre el conjunto de posibles muestras X, de manera que para todo ejemplo $x \in X$, el clasificador es capaz de asignarle una clase $f(x) = y \in Y$ [52].

En algunas ocasiones, se incluye una clase adicional y_0 dentro del conjunto de clases Y, para que el clasificador etiquete las muestras a las que no puede asignarle con certeza alguna de las clases iniciales. En este caso, al conjunto $Y \cup \{y_0\}$ se le denota como Y^*. En otras

ocasiones, el clasificador no asigna una clase directamente, sino la probabilidad de pertenecer a una clase particular. En este caso, al clasificador se le puede ver como una función de la forma $f : X \times Y \rightarrow [0, 1]$.

Entre los algoritmos de clasificación más destacados se encuentran: los árboles de decisión [64], la regresión logística [65] y las máquinas de vector soporte [66].

Problemas de regresión

En los problemas de regresión, se construye un modelo que relaciona las variables de un conjunto de datos de entrada, conocidas como variables predictoras, con una variable numérica que toma valores dentro de un rango continuo, denominada variable respuesta. El modelo desarrollado puede aplicarse para predecir valores de la variable respuesta a partir de las variables predictoras en nuevos datos de entrada. Un campo de aplicación que requiere la predicción de variables continuas es la predicción de series temporales, como se verá en la sección 2.7.

Un tipo de modelo comúnmente utilizado en los problemas de regresión es el modelo de regresión lineal. Formalmente, este tipo de modelos se expresa de la siguiente manera: suponiendo que la variable respuesta se denota como Y, y las p variables predictoras como $X_1, X_2, \ldots, X_p$, un modelo de regresión lineal adopta la forma:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_p X_p + \epsilon$$

donde $\alpha, \beta_1 \cdot \cdot, \beta_p$ representan parámetros desconocidos llamados coeficientes de regresión. Estos parámetros se ajustarán automáticamente a partir de los ejemplos obtenidos del conjunto de entrenamiento. El término ϵ en esta relación implica que la variable respuesta está sujeta a variabilidad y no puede ser completamente expresada a partir de las variables predictoras. Algunos algoritmos comúnmente empleados en este tipo de aprendizaje son los árboles de clasificación y regresión CART [67], regresión no lineal [68], regresión lineal bayesiana [69] y regresión polinómica [70].

Para que el aprendizaje automático sea correcto, entendiéndose como un proceso de generalización a partir de muestras concretas, se necesita disponer de suficientes casos de entrenamiento. De lo contrario, el modelo aprenderá de manera inexacta, y sus resultados no serán fiables en la práctica.

2.2. Redes Neuronales Artificiales

Las *redes neuronales artificiales* (RNA) son modelos matemáticos o computacionales compuestos por unidades de procesamiento numérico conectadas entre sí [71]. Cada unidad de procesamiento se conoce como una neurona artificial. Mediante la manipulación de las conexiones entre las neuronas, se logra que las redes tengan el comportamiento deseado para ser útiles en la solución de problemas complejos, incluidos los problemas de clasificación y regresión.

2.2.1. Inspiración biológica

Las RNA se inspiran en las redes neuronales de tipo biológico [72]. Una neurona es una célula fundamental del sistema nervioso cuya función principal es recibir, integrar, generar y transmitir impulsos eléctricos y químicos para generar actividad cerebral [73]. Morfológicamente, estas células están constituidas por el soma o cuerpo celular, unas prolongaciones cortas por donde ingresa la información, denominadas dendritas, y el axón, una prolongación larga que conduce los pulsos eléctricos hacia otras neuronas. La figura **2-1** muestra esquemáticamente las partes de una neurona.

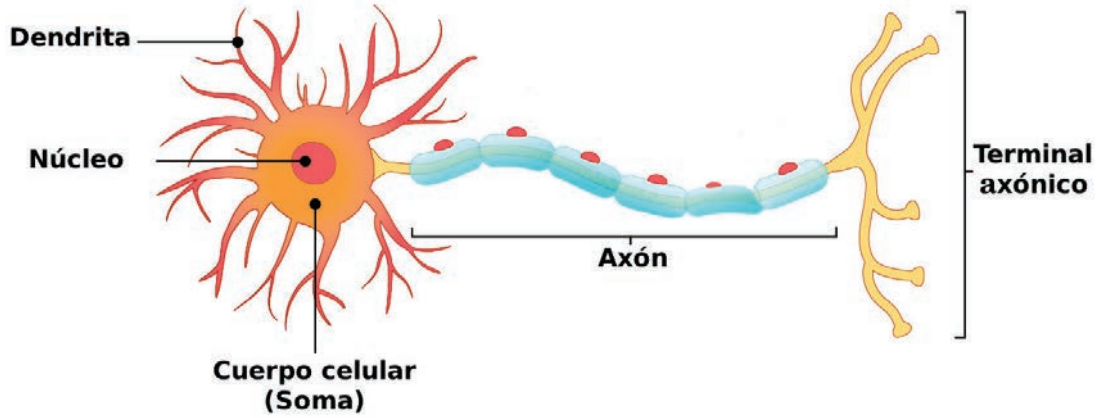


Figura 2-1.: Partes de una neurona biológica.

Los modelos computacionales que se utilizan en la IA prescinden de detalles que son relevantes para la neurociencia, como el retardo de propagación de la señal a lo largo de un axón o las dimensiones físicas de las neuronas (su tridimensionalidad). Un modelo de neurona biológicamente plausible, el cual sirve de base para construir una red neuronal artificial, es el propuesto por los fisiólogos y biofísicos ingleses Hodgkin y Huxley [74], que se explica a continuación.

El fluido extracelular que rodea a la neurona contiene una alta concentración de iones de sodio Na^+ [75]. La propagación de un pulso eléctrico a lo largo del axón, conocido como potencial de acción, depende de la permeabilidad selectiva de la membrana celular de la neurona a estos iones y a otros presentes en su entorno. La membrana celular grasa actúa como aislante eléctrico para la neurona, comportándose de manera similar a un condensador eléctrico [76]. En su estado de reposo, la neurona se encuentra polarizada negativamente. Hodgkin y Huxley observaron que, para generar un potencial de acción, la neurona permite el ingreso de iones Na^+ a través de su membrana, desencadenando un proceso de despolarización. Cuando el potencial dentro de la neurona alcanza un valor umbral, se produce el potencial de acción. Durante este proceso, también se permite la entrada de iones de potasio

K^+, lo que intensifica la corriente generada por la neurona. Una vez finalizado el pulso, la membrana celular permite el paso más lento de iones Na^+, K^+ y otros iones menos relevantes, permitiendo que la neurona regrese a su estado de reposo.

Los trabajos de Hodgkin y Huxley les llevaron a proponer un circuito eléctrico equivalente al de la membrana neuronal, como el que se muestra en la figura **2-2**, aquí, R_{Na}, R_K, R_R indican la resistencia al paso de la corriente de iones de Sodio I_{Na}, potasio I_K y de otros iones presentes en su entorno I_R. Por su parte, V_{Na}, V_K, V_R representan los los respectivos potenciales iónicos.

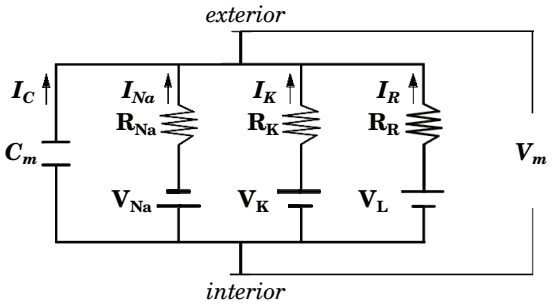


Figura 2-2.: Circuito equivalente al de una membrana con permeabilidad al Na^+, K^+ y a otros iones R.

Aplicando las leyes de Kirchhoff [77] en el circuito equivalente, Hodgkin y Huxley proponen el modelo del potencial de la membrana V_M como:

$$C_M \frac{dV_M}{dt} + I_{Na} + I_K + I_R = 0, \tag{2-1}$$

donde C_M es la capacidad asociada a la membrana como condensador.

Si se representa la permeabilidad selectiva de la membrana como la conductividad para los respectivos iones, g_{Na}, g_K y g_R, donde la conductividad es el inverso de la resistencia, las intensidades respectivas se pueden expresar como:

$$\begin{aligned} I_{Na} &= g_{Na}(V_M - V_{Na}), \\ I_K &= g_K(V_M - V_K), \\ I_R &= g_R(V_M - V_R), \end{aligned} \tag{2-2}$$

A partir de las igualdades (2-2), la ecuación (2-1) se puede reescribir como:

$$C_M \frac{dV_M}{dt} = -g_{Na}(V_M - V_{Na}) - g_K(V_M - V_K) - g_R(V_M - V_R). \tag{2-3}$$

A nivel de la IA, no se necesita un modelo con este nivel de detalle propio del campo de la neurociencia. Para trabajar con un mayor nivel de abstracción, se omite la consideración

de la permeabilidad de los iones Na^+ y K^+, con lo cual se obtiene el modelo de integración y disparo con pérdidas, en inglés: leaky integration-and-fire neuron model [78, 52, 79]:

$$C_M \frac{dV_j}{dt} = -g_R(V_j - V_R) + g_{int} \sum_{i=1}^{n} x_i w_{ij}, \tag{2-4}$$

donde V_j representa el potencial de la membrana para la neurona j, x_i corresponde a la salida actual de la neurona $i-$ésima, w_{ij} indica el peso de la conexión de la neurona i con la neurona j, y n, es el número total de neuronas cuyos impulsos eléctricos llegan a la neurona j. En esta ecuación g_{int} y g_R son dos conductancias constantes, la primera es la constante de integración y la segunda la constante de perdida.

Si asumimos que el potencial generado por la presencia de los otros iones que intervienen en el potencial de acción es despreciable, $V_R = 0$, y que los factores de escala asociados a las conductancias son iguales, $g_{int} = g_R = 1$, se obtiene una expresión más sencilla:

$$C_M \frac{dV_j}{dt} = -V_j + \sum_{i=1}^{n} x_i w_{ij}. \tag{2-5}$$

En un modelo aplicable a la IA, se ignora la corriente de perdida $-V_j$, con lo que se obtiene el modelo de integración y disparo, conocido en inglés como: *the integrate and fire model* [80]. Dado que en este modelo las neuronas tienden a saturarse, una vez eliminada la corriente de perdida que descargaría el condensador, se reemplaza por una anulación de su carga después de cada intervalo de tiempo. De esta manera, el voltaje de salida de la neurona depende únicamente de la suma ponderada de sus pesos y entradas. Desde el punto de vista biológico, este modelo representa una neurona sin memoria, ya que en ningún momento interviene el valor del potencial eléctrico de la neurona en el instante de tiempo anterior:

$$V_j = \sum_{i=1}^{n} x_i w_{ij}. \tag{2-6}$$

La ecuación (2-6) representa el modelo comúnmente utilizado como base para construir redes neuronales artificiales. En este modelo, la salida de la neurona está determinada por la combinación de las entradas a la neurona multiplicadas por sus respectivos pesos. Habitualmente, se incluye un sesgo o bias en la entrada de la neurona, añadiendo un peso adicional $w_{0j} = b_j$ relacionado con una entrada fija con valor 1, $x_0 = 1$. Esta inclusión del sesgo permite obtener finalmente el modelo para una neurona artificial, expresado como:

$$V_j = \sum_{i=0}^{n} x_i w_{ij}. \tag{2-7}$$

La figura **2-3** representa gráficamente la ecuación (2-7).

Como se verá más adelante, las RNA están compuestas por capas. Cada capa contiene n neuronas que comparten características entre sí. Cada una de las n neuronas de la capa se denota como x_i. Los pesos w_{ij} se utilizan para modelar las conexiones de entrada a las neuronas, refiriéndose a la conexión entre la salida de la neurona $i-$ésima de una capa y la entrada de la neurona $j-$ésima de otra capa. La salida de cada neurona, representada

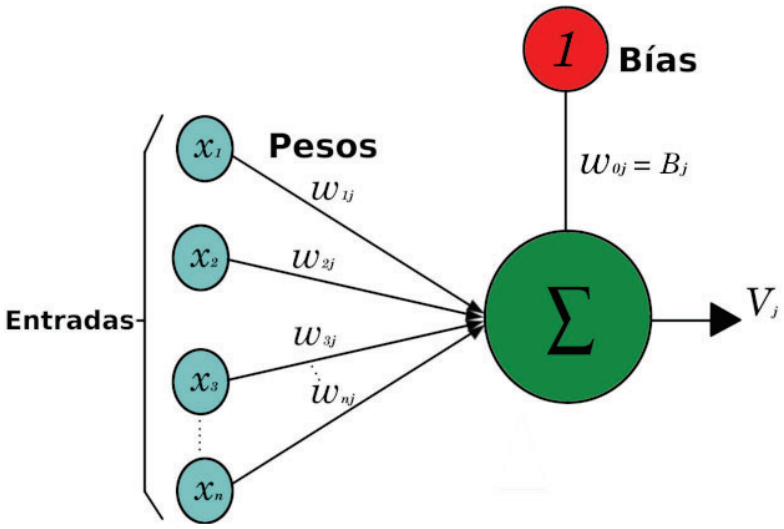


Figura 2-3.: Modelo simplificado de una neurona artificial.

como y_i, se obtiene al aplicar una función f a la combinación de sus entradas multiplicadas por sus pesos, permitiendo, de este modo la generación de valores de salida que cumplan ciertas condiciones deseadas (salidas lineales, no lineales, continuas, discretas, acotadas, no acotadas etc.)

$$y_j = f\left(\sum_{i=0}^{n} x_i w_{ij}.\right). \tag{2-8}$$

En el campo de las RNA, a las funciones f se les denomina *funciones de activación*, ya que determinan si la neurona activa o no su salida. Las funciones de activación se pueden clasificar como discretas o continuas. En el caso de las funciones discretas, se utilizan típicamente valores como 0 o 1, o incluso -1 o 1, para la salida. En el primer caso, se habla de neuronas binarias, mientras que en el segundo caso, de neuronas bipolares. Por otro lado, las funciones de activación continuas normalmente tienen un rango de salida en el intervalo $[0, 1]$ ó $[-1, 1]$. Dentro de este conjunto, se encuentran tanto las funciones lineales como las no lineales. Las funciones no lineales son las más comúnmente utilizadas.

2.2.2. Funciones de activación

Existen varias funciones de activación empleadas en modelos de neuronas artificiales; en este trabajo se describen las funciones lineal, lineal rectificada y tangente hiperbólica, que han sido las aplicadas en el estudio realizado. Una explicación más detallada de estas y otras funciones de activación se puede encontrar en [52, 81].

Función de activación lineal

Una función de activación lineal [81] simplemente toma una entrada x y produce una salida $c \cdot x$, donde c representa una constante, figura **2-4**.

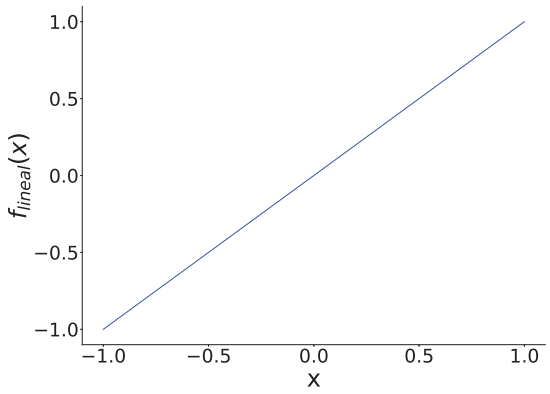


Figura 2-4.: Función de activación lineal.

En el caso de $c = 1$, la función resultante es la identidad. Es importante destacar que, en una Red Neuronal Artificial RNA, tener dos o más capas lineales interconectadas entre sí es equivalente a tener una sola capa lineal. Esto se debe a que la última capa continuará siendo una función lineal de la primera. Para demostrar este hecho, basta suponer una capa de neuronas con una matriz de pesos W_1, cuya salida está conectada a otra capa de neuronas lineales con pesos W_2. El resultado de esta composición es:

$$y = f_2(f_1(x)) = W_2(W_1x) = (W_2W_1)x, \tag{2-9}$$

indicando que se pueden substituir ambas capas por una única capa de pesos $W = W_2W_1$.

Una RNA compuesta únicamente por elementos lineales solo puede emplearse para representar funciones lineales. Esto descartaría su uso práctico para resolver problemas reales, ya que la gran mayoría presentan algún tipo de no linealidad.

Función de activación lineal rectificada

La función lineal rectificada, ReLU por sus siglas en inglés [82], se define como: [1]

$$f_{relu}(x) = \begin{cases} 0 & si \quad x \leq 0, \\ \\ x & si \quad x > 0. \end{cases} \tag{2-10}$$

Es la función de activación más popular en redes neuronales [83, 84]; su representación gráfica se puede observar en la figura **2-5**.

Una característica importante es que aporta un carácter no lineal y está construida sin el uso de funciones trascendentales. Debido a que no requiere operaciones aritméticas complejas para determinar su salida, se considera una función con una gran velocidad de convergencia [85]. Además, su derivada es trivial, ya que equivale a la función escalón (2-11).

[1] En varios textos, suele definirse también como: $f_{relu}(x) = max(0, x)$

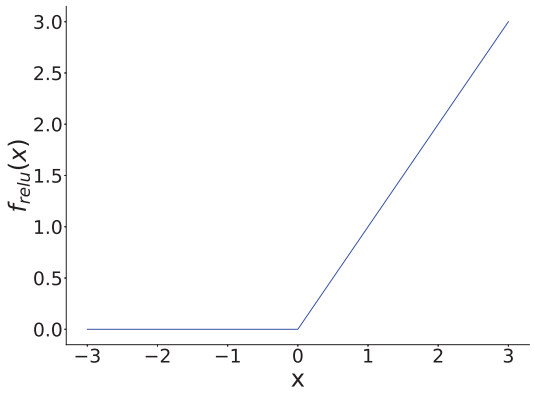


Figura 2-5.: Función de activación ReLU.

$$f'_{relu}(x) = \begin{cases} 0 & si \quad x \leq 0, \\ \\ 1 & si \quad x > 0. \end{cases} \tag{2-11}$$

Como se puede observar, esta función asume que su derivada en el punto cero, es cero. Generalmente, este supuesto no suele ser un inconveniente y funciona bien en la mayoría de los casos [48].

Función de activación tangente hiperbólica

La función de activación tangente hiperbólica (tanh) [86], así como las funciones logística [87] y gudermanniana [88], forman parte del conjunto de funciones sigmoideas [89]. Todas estas funciones exhiben una forma característica de 'S', lo que las convierte en funciones no lineales. Son funciones estrictamente crecientes, continuas y derivables, propiedades matemáticas que las hacen especialmente interesantes para su uso en redes neuronales artificiales [52].

La función tangente hiperbólica se define como:

$$f_{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}. \tag{2-12}$$

Esta función se representa gráficamente en la figura **2-6**:

La derivada de esta función,

$$f'_{tanh}(x) = (1 - tanh^2(x)), \tag{2-13}$$

permite que su valor en un punto pueda ser expresado en términos de la misma función en ese punto.

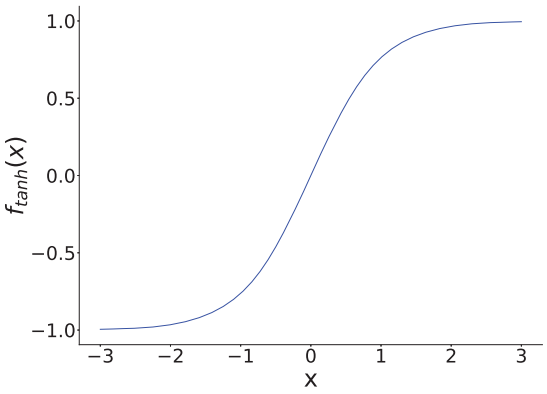


Figura 2-6.: Función de activación tanh.

2.3. Arquitectura de una red neuronal

Las neuronas artificiales, por sí solas, tienen un uso limitado, por lo que es necesario considerar agrupaciones o colecciones de neuronas interconectadas. A una colección de neuronas similares que operan conjuntamente, se le conoce como una capa de la red. Cada combinación posible entre las capas, dan lugar a un tipo de arquitectura de red distinta que suele emplearse para referirse al tipo de red utilizada. Entre las arquitecturas más populares a la fecha se distinguen las redes neuronales de propagación hacia adelante (*feed-forward neural networks*) [90], las redes neuronales competitivas (*competitive neural networks*) [91], las redes recurrentes (*recurrent neural networks*) [92], las redes neuronales convolucionales (*convolutional neural networks*) [93] , las redes generativas adversarias (*generative adversarial network*) [94] y las redes neuronales de tipo transformer (transformer neural networks) [95], entre otras. A continuación, se describe brevemente las tres primeras arquitecturas. Para una descripción más amplia de las arquitecturas restantes y sus aplicaciones, se recomienda consultar [96]

2.3.1. Redes de propagación hacia adelante

La topología más habitual en las redes neuronales corresponde a las redes de propagación hacia adelante, también conocidas como *redes feed-forward*. Estas redes generalmente se componen de múltiples capas, divididas en dos categorías: visibles y ocultas. Una capa se etiqueta como visible si ocupa la primera o última posición en la red; en caso contrario, se considera una capa oculta. La primera capa se denomina capa de entrada, mientras que la última se designa como capa de salida. En esta arquitectura, las neuronas de cada capa tienden a ser independientes y operan en paralelo.

En esta estructura, las neuronas de la capa de entrada no funcionan como neuronas artificiales en el sentido estricto de la palabra. Su función principal es recibir y transmitir patrones a todas las neuronas de la siguiente capa. Por otro lado, la capa de salida devuelve al exterior la respuesta de la red neuronal para cada patrón de entrada. Las neuronas de las capas

ocultas son responsables de realizar el procesamiento no lineal de los patrones recibidos.

Como se ilustra en la figura **2-7**, las conexiones en estas redes son unidireccionales, lo que significa que la salida de la capa i actúa como entrada para la capa j, de ahí su nombre. Cada conexión entre neuronas tiene asociado un número real conocido como peso de la conexión. Además, todas las neuronas tienen un sesgo o *bias* en su entrada, que generalmente se trata como una conexión adicional a la neurona, tal como se puede ver en la Sección 2.2.1.

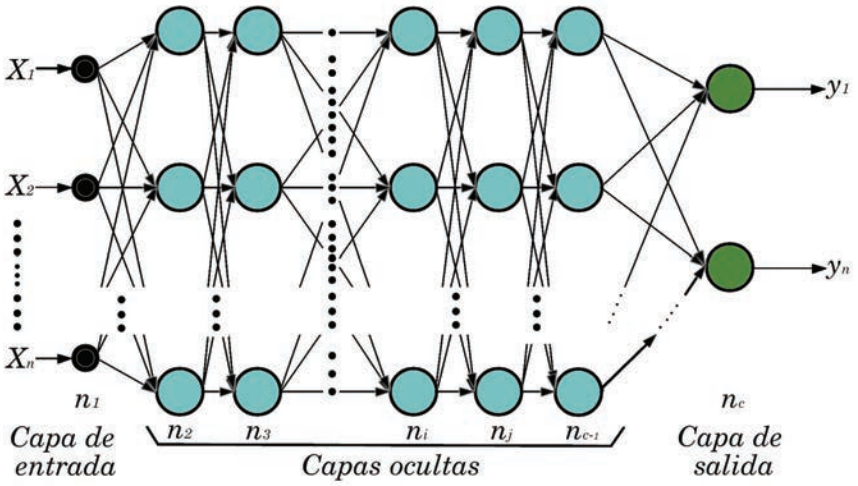


Figura 2-7.: Arquitectura de una red de propagación hacia adelante.

Generalmente, se distingue entre dos tipos de redes neuronales *feed-forward* que pueden estar conectadas de manera total o parcial [97]. En las redes totalmente conectadas, las salidas de una capa se conectan con todas las entradas de la siguiente capa. Por otro lado, en las redes parcialmente conectadas, las salidas de una capa son recibidas por un grupo seleccionado de neuronas de la capa siguiente, lo que permite crear módulos especializados para tareas específicas. Estos dos enfoques tienen aplicaciones diversas. Por ejemplo, se emplean redes parcialmente conectadas para la predicción de tendencias en el precio de las acciones, utilizando indicadores técnicos como datos de entrada [98]. También se utilizan en la detección temprana o de reincidencia del cáncer de mama [99].

Las redes neuronales *feed-forward* pueden ser de tres tipos: simples, multicapa o profundas [52]. Una red simple consta únicamente de una capa de entrada y una capa de salida. En este tipo de redes, la capa de entrada recibe la información y la transmite a las neuronas de la capa de salida, donde se procesa. Un ejemplo clásico de este tipo de red es el perceptrón, desarrollado en 1943 por Warren McCulloch y Walter Pitts [100].

Las redes neuronales multicapa se caracterizan por incluir una capa oculta entre la capa de entrada y la capa de salida. Estas redes tienen la capacidad de ser aproximadores universales si se incluyen funciones de activación sigmoideas, como lo demostraron George Cybenko [101] y Ken-ichi Funahashi [102] independientemente. Esto implica que, al usar una sola capa oculta con un número finito de neuronas, se dota a una red neuronal de la capacidad de aproximar funciones continuas no lineales en subconjuntos compactos de $\mathbb{R}^n$. Posteriormente, Kurt Hornik mostró que las derivadas de la red multicapa pueden aproximar las derivadas

de la función aproximada por la red neuronal [103]. Además, demostró que basta con que las funciones de activación sean continuas, no constantes y acotadas, para que una red neuronal multicapa funcione como un aproximador universal [104]. Aunque se requiere cierta no linealidad en las funciones de activación según sus trabajos, son las propiedades de la topología de la red las que le confieren su carácter de aproximador universal a una red multicapa, y no necesariamente la elección específica de la función de activación [52].

Por otro lado, las redes neuronales profundas, también conocidas como Deep Neural Networks, se caracterizan por tener más de una capa oculta en su arquitectura. Al contar con múltiples capas, no solo se convierten en aproximadores universales, sino que, además, pueden aprender mediante la técnica de aprendizaje por representaciones. Esto significa que son capaces de construir una estructura jerárquica al extraer características complejas a partir de datos de entrada más simples. Esta capacidad les permite identificar patrones relacionados con las salidas deseadas. Las redes neuronales profundas destacan especialmente en tareas como el reconocimiento de objetos en imágenes [105], el reconocimiento del habla [106] y la predicción de series temporales [107], entre otras.

2.3.2. Redes competitivas y redes recurrentes

Una red neuronal competitiva incluye conexiones inhibitorias entre las neuronas de una misma capa, de manera que si una neurona se activa, ejerce una influencia inhibidora en las otras neuronas de su capa [108]. Estas redes suelen emplearse en problemas donde una capa debe responder en función del contexto, como en el reconocimiento del habla [109], compresión y segmentación de imágenes [110, 111] o reducción de dimensionalidad en datos [112].

Por otro lado, en las redes neuronales recurrentes, se permite el flujo de información de las neuronas de una capa con las neuronas de las capas siguientes y anteriores [52]. Este flujo de información bidireccional dota de memoria a estas redes neuronales, lo que las hace útiles en tareas como el reconocimiento no segmentado y conectado de la escritura [113], el reconocimiento del habla [114] y la predicción de series temporales [115, 116, 117].

2.4. Entrenamiento de una red feed-forward

En este trabajo, se emplearon redes neuronales *feed-forward* completamente conectadas para desarrollar una versión híbrida de un propagador orbital del tipo Encke. A continuación, se describen los pasos para entrenar una red neuronal con esta arquitectura, proporcionando algunos detalles técnicos. Para obtener una visión más general sobre el entrenamiento de una red neuronal con otras arquitecturas, se puede consultar [48, 52]

2.4.1. Topología de la red

Antes de entrenar una red neuronal, es crucial definir el número de capas y el número de neuronas en cada una de ellas. En principio, parece ser una tarea sencilla, pues de acuerdo con lo mencionado en la sección 2.3.1, bastaría con una red multicapa, con un número

suficiente de neuronas y de muestras, para construir un modelo capaz de representar los patrones que pueden contener los datos de entrenamiento. El problema se encuentra en que Cybenko y Funahashi solo demuestran la existencia de la red, pero no determinan cuál es ese número suficiente de neuronas ni cómo se puede calcular. Además, desde el punto de vista computacional, esta metodología puede ser ineficiente, ya que, para aproximar algunas funciones, se requiere que la red multicapa tenga un número muy grande de neuronas en la capa oculta [118, 119]. Sin embargo, se demostró [120] que, al emplear funciones de activación lineales a trozos, una red neuronal con n capas ocultas puede representar eficientemente funciones que no podrían ser representadas por una red de $n-1$ capas ocultas, sin incrementar exponencialmente el número de neuronas ocultas. Este resultado se aplica también a las redes multicapa ($n = 1$). El éxito de las redes profundas sobre las redes multicapa radica en su capacidad para descomponer problemas complejos de manera jerárquica, lo que se alinea perfectamente con su habilidad para extraer características complejas a partir de características más simples (*Representation Learning*).

En la práctica, comúnmente se inicia el proceso de entrenamiento con una red neuronal multicapa. Si la red neuronal no converge durante el entrenamiento, se puede considerar la adición de neuronas a la capa oculta (si no es posible aumentar el conjunto de datos de entrenamiento) hasta lograr el rendimiento deseado [52]. Por otro lado, si la red converge durante el entrenamiento y se busca reducir la carga computacional, existen técnicas de poda disponibles para eliminar conexiones poco significativas en la generación de la respuesta de la red [121]. En relación al número óptimo de capas en la red, se sugiere añadir capas ocultas hasta que el error en los datos de validación deje de disminuir [122]. Los datos de validación constituyen un conjunto de datos reservados para evaluar el desempeño de la red neuronal durante su entrenamiento.

2.4.2. Inicialización de los pesos de la red

Una vez definida la arquitectura inicial para la red neuronal, el siguiente paso es establecer el algoritmo que va a inicializar los pesos entre las conexiones de la red. A menudo, esta etapa no recibe la atención necesaria, a pesar de que una inicialización adecuada de los pesos es crucial para lograr la convergencia del modelo durante el proceso de entrenamiento. Por regla general, se inicializan los pesos de manera aleatoria con valores pequeños extraídos de una distribución uniforme o normal [52]. Esta aproximación busca evitar la saturación de las funciones de activación. La saturación hace referencia a un estado que perjudica el aprendizaje de la red neuronal, caracterizado porque una neurona emite predominantemente valores cercanos a los extremos asintóticos de su función de activación [123]. Sin embargo, si los valores de inicialización son muy pequeños, es posible que se anulen durante el proceso de entrenamiento, llevando a la red neuronal a dejar de aprender. Esto sucede porque, al ser los pesos muy similares entre sí, cada actualización los modifica en la misma cantidad, manteniéndolos prácticamente iguales durante todo el proceso.

Con el fin de evitar tanto la saturación de las funciones de activación como la anulación de los pesos de la red, se han desarrollado diferentes estrategias de inicialización de pesos. Entre las más conocidas se encuentran el método de inicialización de Xavier [124], el de Lecun [125] y el de He [126].

2.4.3. Actualización de los pesos de la red

Una vez que los pesos de la red neuronal están inicializados, se procede a ajustarlos para que la salida de la red sea lo más cercana posible a la salida requerida. Este proceso se lleva a cabo utilizando un algoritmo de entrenamiento supervisado, formulando el aprendizaje de la red a través de un problema de optimización:

$$W^* = \arg\min E(W),$$

donde W es el conjunto de los pesos de la red neuronal, E es una función de error o coste que evalúa la diferencia entre las salidas de la red y las salidas requeridas. W^* representa entonces, el conjunto de pesos que se pasan como argumento a la función E, para que consiga el mínimo valor ($\arg\min E$). En el Anexo B se describen las funciones de coste empleadas con mayor frecuencia en el entrenamiento de las RNA.

Al emplearse funciones de activación no lineales, la respuesta de la red también es no lineal, lo que implica la necesidad de emplear técnicas de optimización no lineales. Estas técnicas se basan, en gran medida, en la actualización de los pesos siguiendo la dirección negativa del gradiente de la función de error o coste E. Los métodos de optimización basados en el gradiente son los más utilizados en el entrenamiento de las redes neuronales debido a su capacidad para abordar problemas de optimización en espacios multidimensionales. Al utilizar uno de estos métodos, el problema se plantea de la siguiente manera:

$$\Delta w = -\eta \nabla E. \tag{2-14}$$

Esta expresión en notación vectorial, se traduce en el cálculo de las derivadas parciales del error con respecto a cada uno de los pesos de la red:

$$\Delta w_{ij}^{C} = -\eta \frac{\partial E}{\partial w_{ij}^{C}}.$$

La variable η representa la tasa de aprendizaje, conocida en inglés como *Learning Rate* (LR), que es un parámetro que controla el tamaño de las actualizaciones que se realizan sobre los pesos de la red neuronal; w_{ij}^{C} es el peso de la conexión entre la neurona i en la capa C y la neurona j en la capa $C+1$ de la red neuronal.

La estimación del gradiente se realiza a través de métodos numéricos.[2] Como en todo método numérico, dicha estimación acumula los errores de aproximación, lo que genera problemas de convergencia en el entrenamiento de la red neuronal. Debido a que la tasa de aprendizaje regula cuánto se ajustan los pesos en cada iteración, resulta un parámetro fundamental para conseguir un buen desempeño del algoritmo de optimización. Por ejemplo, si la tasa de aprendizaje es muy pequeña, la convergencia del algoritmo de entrenamiento de la red puede tardar más de lo necesario. En caso contrario, si la tasa de aprendizaje es muy elevada, puede generar comportamientos inestables del error, impidiendo la convergencia del entrenamiento.

[2]Salvo en contadas excepciones, que se puede realizar a través de métodos analíticos.

Dada la importancia de la tasa de aprendizaje, se han propuesto numerosos métodos para realizar automáticamente un ajuste adaptativo de este parámetro. Varios de estos métodos se encuentran integrados dentro de plataformas de Machine Learning como Tensorflow 2.0 y Pytorch. Entre los más comunes, se pueden destacar los métodos: *AdaGrad* [127], *AdaDelta* [128], *Rprop* [129] y *Adam* [130].

2.5. Ajuste de hiperparámetros

Los hiperparámetros de una red son aquellos parámetros que determinan su estructura y que no se modifican durante el proceso de entrenamiento. Por ejemplo, las funciones de activación, los parámetros que definen la topología de la red y los parámetros que configuran los algoritmos empleados. Para obtener buenos resultados durante el entrenamiento de una red neuronal, se debe lograr una buena sincronía entre todos los hiperparámetros. Desafortunadamente, de momento no existe una técnica analítica que indique la dirección en la que se debe buscar la mejor combinación, como sucede en el ajuste de los pesos durante el entrenamiento de la red. Las técnicas de ajuste que buscan una combinación óptima de hiperparámetros, y que en la práctica pueden generar resultados satisfactorios, se pueden dividir en dos categorías: *técnicas de ajuste manual* y *técnicas de ajuste automático*. Las técnicas de ajuste manual se basan en la experiencia del investigador. Por lo general, se comienza creando un modelo sencillo de una red neuronal mínimamente funcional y se lleva a cabo un proceso iterativo de prueba y error. Este procedimiento permite realizar las variaciones necesarias en cada paso para encontrar las combinaciones de hiperparámetros que proporcionen el mejor desempeño posible. Dentro de los métodos automáticos de búsqueda de hiperparámetros, o en inglés *hyperparameters search*, se encuentran el método sistemático, el método de búsqueda aleatoria y los métodos de búsqueda inteligente [52]. A continuación, se describirá brevemente cada uno de ellos. Es importante mencionar que plataformas de Machine Learning como Tensorflow 2.0 y PyTorch cuentan con librerías que implementan los algoritmos más conocidos para la búsqueda de hiperparámetros, incluyendo las tres técnicas que serán descritas en esta sección. Entre estas librerías se destacan Hyperopt [131], Spearmint [132] y DNGO [133].

2.5.1. Búsqueda sistemática

La búsqueda sistemática, o en inglés *grid search*, implica evaluar todas las combinaciones posibles de los hiperparámetros preseleccionados. En esta búsqueda, se genera un conjunto de k posibles valores para cada hiperparámetro. Al cruzar los valores entre los distintos hiperparámetros se crea una rejilla o *grid*, de ahí su denominación. El método recorre sistemáticamente la rejilla evaluando cada una de las posibles configuraciones. Es importante destacar que este método es computacionalmente costoso, especialmente cuando se define un número elevado de hiperparámetros, ya que deben evaluarse las combinaciones de todos los posibles valores de cada hiperparámetro con todos los valores de cada uno de los demás hiperparámetros. No obstante, es el único en garantizar la exploración exhaustiva de todas las configuraciones posibles para encontrar la mejor combinación de hiperparámetros.

2.5.2. Búsqueda aleatoria

En la búsqueda aleatoria, también conocida como *random search* en inglés, se seleccionan aleatoriamente algunas combinaciones de hiperparámetros de la rejilla. Su principal ventaja es que tarda un tiempo significativamente menor en encontrar una buena combinación de hiperparámetros en comparación con la búsqueda sistemática. Sin embargo, su principal desventaja es que no garantiza la identificación de la mejor combinación posible.

2.5.3. Búsqueda inteligente

En la búsqueda inteligente, o por su denominación en inglés *smart search*, se orienta el proceso para que resulte más eficiente en la exploración del espacio de configuraciones de los hiperparámetros, en comparación con la simple búsqueda aleatoria. Normalmente, estos algoritmos buscan encontrar la mejor configuración combinando dos procesos: exploración y explotación. En el primero, los algoritmos exploran posibles combinaciones de hiperparámetros que podrían generar buenas arquitecturas de red, aunque con una mayor incertidumbre. En el segundo paso, explotan configuraciones que previamente mostraron un rendimiento aceptable y con una alta certeza de poder mejorar con pequeñas modificaciones.

2.6. Sobreaprendizaje

El sobreaprendizaje, conocido en inglés como *overfitting*, es un comportamiento no deseado del aprendizaje automático supervisado. Ocurre cuando el modelo se ajusta al ruido o la varianza de los datos en lugar de aprender la distribución subyacente de la que fueron extraídos [134]. Un modelo sobreajustado puede ofrecer predicciones precisas sobre la muestra de datos de entrenamiento, pero su precisión se reduce significativamente al enfrentarse a datos desconocidos. Generalmente, el sobreajuste aparece cuando el conjunto de entrenamiento es muy pequeño, contiene mucho ruido o no representa adecuadamente la diversidad de los datos.

Para prevenir el sobreajuste en las redes neuronales, se dispone de varias estrategias [135]. Aumentar el tamaño del conjunto de datos de entrenamiento es una de ellas, ya que permite incrementar las características presentes en el conjunto de entrenamiento, proporcionando a la red más información sobre la distribución de los datos. La combinación de múltiples modelos neuronales es otra estrategia que busca compensar los errores de una red con los aciertos de otras. Una técnica comúnmente usada, conocida en inglés como *model averaging*, consiste en promediar las predicciones realizadas por cada una de las redes entrenadas. Otra técnica, denominada en inglés *Bayesian fitting*, implica combinar las predicciones de varias redes neuronales entrenadas con una única arquitectura. El sobreajuste, producto de la alta flexibilidad del modelo que se adapta demasiado bien a los detalles y particularidades de los datos de entrenamiento, también puede ser controlado mediante una técnica llamada *regularización*.

La regularización hace referencia a cualquier técnica que modifique el algoritmo de aprendizaje de un modelo para reducir su error de generalización sin incrementar su error en el conjunto de los datos de entrenamiento. Algunas técnicas de regularización están diseñadas

para modificar la función de coste, introducir restricciones sobre los parámetros de la red o añadir ruido durante el proceso de entrenamiento [136]. Sin embargo, entre estas técnicas, una de las más recomendadas es la conocida como *parada temprana*, la cual se describe a continuación.

2.6.1. Parada temprana

La parada temprana, más comúnmente conocida como *early stopping* [137], se basa en el uso de los datos de validación para generar una señal que indique al algoritmo de aprendizaje que la red neuronal ha dejado de aprender y ha comenzado a sobreajustarse. Una vez es recibida la señal, el algoritmo detiene el proceso de entrenamiento. El concepto detrás del early stopping es sencillo pero sumamente poderoso. En principio, si el entrenamiento de la red está convergiendo, los errores en los datos de entrenamiento y validación deben disminuir en cada iteración del algoritmo. Sin embargo, cuando la red neuronal comienza a ajustarse excesivamente a los datos de entrenamiento, empieza a perder su capacidad de generalización. Como resultado, el error en los datos de validación comienza a aumentar. En este punto, se envía la señal para detener el aprendizaje y se conservan los valores de los parámetros correspondientes al punto donde se obtuvo el mínimo error en el conjunto de validación.

En la práctica, no es usual detener el entrenamiento justo después de detectar un incremento en el error en los datos de validación, sino que se suele dar un margen de espera. Este margen es un parámetro constante conocido como la *paciencia* del algoritmo. La paciencia, o por su denominación en inglés *patience*, indica al algoritmo cuántas iteraciones debe esperar sin ver mejoras en el error sobre el conjunto de datos de validación antes de dar la señal de detener el entrenamiento.

El early stopping proporciona un mecanismo efectivo de regulación, sin tener que manipular la función de coste o incluir restricciones sobre los parámetros de la red que puedan perjudicar el aprendizaje. Además, es perfectamente compatible con otras técnicas de regularización como el dropout [138] o la introducción de ruido [139].

2.7. Predicción de series temporales discretas univariadas

Una serie temporal discreta univariada (en adelante, simplemente serie temporal) es una colección de observaciones ϵ_t, que se registran en tiempos t procedentes de un conjunto discreto y ordenado T_0 [140]. Un ejemplo de una serie temporal se puede ver en la figura **2-8**, donde se registra la demanda mensual de electricidad en Gigavatios hora (GWh) en el estado de Victoria, en Australia, para el periodo de enero de 2012 a diciembre de 2014, $T_0 = \{1, 2, 3...156\}$. [3]

Las redes neuronales son ampliamente empleadas en la predicción de series temporales debido a su capacidad de aproximar cualquier función, construir relaciones no lineales a partir de información incompleta o con ruido, y por su facilidad de construcción. A continuación, se describe la forma en la que se emplea una red neuronal profunda para predecir el comportamiento de una serie temporal.

[3] Los datos se pueden encontrar en: www.cienciadedatos.net

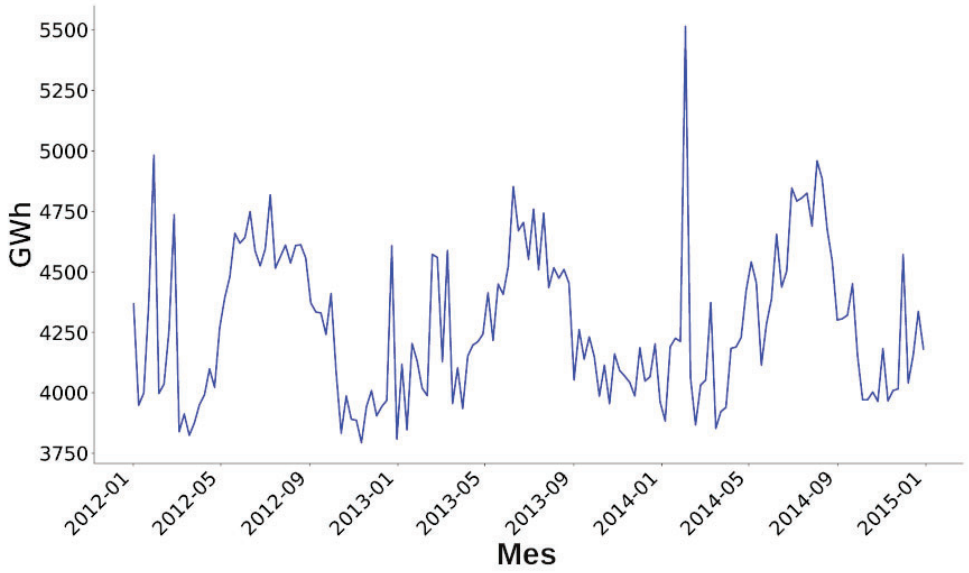


Figura 2-8.: Consumo mensual de energía en Victoria (Australia) en el periodo de 2012 a 2014.

2.7.1. DL para la predicción de series temporales

Las redes neuronales feed-forward son consideradas modelos estáticos, debido a que están diseñadas para encontrar relaciones entre unas variables de entrada y una o varias variables de salida sin tener en cuenta la temporalidad de los datos. Por ejemplo, en el caso de la regresión lineal. No obstante, esto no impide que se puedan utilizar para tratar información temporal y ser aplicadas al problema de la predicción de series temporales. Para ello, se necesita que las variables de entrada a la red contengan, además del valor en un determinado instante de tiempo, información de su comportamiento representado mediante una secuencia finita de datos. Una red neuronal se puede entrenar para predecir uno o varios pasos en cada iteración. En esta investigación, las predicciones del comportamiento de las series temporales se realizan en un paso de tiempo. Para una descripción detallada de las predicciones en múltiples pasos de tiempo, se puede consultar [141].

Predicción en un paso de tiempo

En este caso, se expresa el valor de la serie ϵ_t en el instante de tiempo t, como una función no lineal de los valores de la serie en los $r-1$ instantes anteriores de tiempo:

$$\epsilon_t = F(\epsilon_{t-1}, \epsilon_{t-2}, \epsilon_{t-3}....\epsilon_{t-r+1})) + \mu, \tag{2-15}$$

donde μ es un error residual que se considera como ruido blanco Gaussiano y F es una función no lineal desconocida que debe ser aproximada.

A continuación, la función F se aproxima entrenando una red neuronal con el conjunto de los N datos observados de la serie temporal $\{\epsilon_t\}_{t=1...N}$. Para esto, es necesario formular el problema de la predicción de series temporales como un problema de aprendizaje supervisado. Esto se logra empleando un método para iterar sobre una secuencia de datos, conocido como *ventana deslizante*, normalmente empleado en aprendizaje supervisado y tratamiento de imágenes. Una ventana deslizante es un vector de tamaño fijo r que se desliza por toda la secuencia de datos en pasos de tamaño n. Por ejemplo, si se tiene una secuencia de valores de una serie temporal con $N = 20$ datos, al crear una ventana deslizante de tamaño $r = 5$ y $n = 1$ pasos, se dividen los datos en vectores de 5 valores consecutivos. El primer vector estaría formado por los 5 primeros valores de la serie, luego la ventana se desplaza una unidad, de manera que el segundo vector estaría compuesto por los valores de la segunda a la sexta posición y así sucesivamente. Los primeros $r-1$ valores de cada vector son los datos de entrada a la red neuronal y el último es el dato de salida o la respuesta deseada, como se puede ver en la tabla **2-1**.

	Entrada	Salida deseada
Vector 1	$\epsilon_1, \epsilon_2 ... \epsilon_{r-2}, \epsilon_{r-1}$	ϵ_r
Vector 2	$\epsilon_2, \epsilon_3 ... \epsilon_{r-1}, \epsilon_r$	ϵ_{r+1}
Vector 3	$\epsilon_3, \epsilon_4 ... \epsilon_r, \epsilon_{r+1}$	ϵ_{r+2}
....	...	...
Vector N-(r-1)	$\epsilon_{N-(r-1)} ... \epsilon_{N-2}, \epsilon_{N-1}$	ϵ_N

Tabla 2-1.: Esquema de una ventana deslizante de tamaño r y paso 1, para los datos de una serie temporal de N datos.

Una vez entrenada la red neuronal, puede ser aplicada tanto para reproducir el comportamiento de la serie temporal, como para pronosticar su evolución a partir de nuevos datos de entrada. El pronóstico se hace empleando el método de la ventana deslizante, salvo que una vez se ha calculado el valor de salida de la red a partir del primer vector, la ventana deslizante se desplaza y el nuevo vector de entrada pierde el valor inicial, al tiempo que se añade el valor pronosticado recientemente. Este proceso se prolonga hasta que se alcanza el valor final que se debe pronosticar.

3. Metodología híbrida aplicada al método de Encke

En este capítulo se presenta la metodología híbrida, la cual, permite mejorar la precisión de cualquier tipo de propagador sin incrementar significativamente su carga computacional. La primera sección introduce el concepto de propagador híbrido a partir de un propagador orbital clásico. En la segunda sección, se describe una metodología para el desarrollo de un propagador híbrido que consta de tres etapas. Finalmente, la tercera sección presenta el análisis preliminar destinado a la construcción de un propagador híbrido de tipo Encke basado en el propagador SGP4, especialmente adaptado para la región MEO.

3.1. Propagador híbrido

El objetivo de un propagador híbrido, como el de cualquier otro tipo de propagador, es estimar, de la forma más precisa posible, la posición y la velocidad de un objeto en órbita alrededor de la Tierra o cualquier otro cuerpo celeste $\boldsymbol{x}_t = (x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t)$ en un tiempo final dado t_f, $\widehat{\boldsymbol{x}}_{t_f}$, a partir de la posición y la velocidad en un tiempo inicial t_0, $\boldsymbol{x}_{t_0}$.

Inicialmente, se calcula la posición y la velocidad aproximada del objeto, $\boldsymbol{x}_{t_f}^{\mathcal{P}}$, utilizando un propagador numérico, analítico o semi-analítico $\mathcal{P}$:

$$\boldsymbol{x}_{t_f}^{\mathcal{P}} = \mathcal{P}(t_f, \boldsymbol{x}_{t_0}). \tag{3-1}$$

El propagador $\mathcal{P}$ queda perfectamente determinado por el método de integración y por las perturbaciones que incluye.

A continuación, se utiliza una técnica de predicción que permita modelar las perturbaciones no incluidas en $\mathcal{P}$, mejorar la precisión del método de integración del propagador o, incluso, modelar las incertidumbres intrínsecas del problema. Para ello, es necesario disponer de un conjunto de observaciones precisas, o en su defecto, de pseudo-observaciones generadas por un propagador orbital de alta precisión, $\boldsymbol{x}_{t_i}^{\mathcal{O}}$, para un conjunto discreto de T momentos secuenciales, $\{t_0, ..., t_T : t_T < t_f\}$ denominado *intervalo de control*. Esto es, mediante $\boldsymbol{x}_{t_i}^{\mathcal{O}}$ se representa la dinámica real del objeto, mientras que $\boldsymbol{x}_{t_i}^{\mathcal{P}}$ la calculada por medio del propagador durante el intervalo de control. A partir de estos dos conjuntos de valores, el error del propagador (es decir, la diferencia entre el comportamiento real del objeto y el proporcionado por $\mathcal{P}$) puede determinarse para cualquier instante t_i del intervalo de control como:

$$\boldsymbol{\varepsilon}_{t_i} = \boldsymbol{x}_{t_i}^{\mathcal{O}} - \boldsymbol{x}_{t_i}^{\mathcal{P}}, \tag{3-2}$$

donde el conjunto de seis series temporales, $\boldsymbol{\varepsilon}_{t_i}$, contiene los efectos que le faltan a $\mathcal{P}$ para ajustarse a la dinámica real del objeto. Como el objetivo de esta metodología es pronosticar

dichos efectos fuera del intervalo de control, se utilizarán técnicas estadísticas de predicción o métodos de IA para construir modelos predictivos para las series temporales $\boldsymbol{\varepsilon}_{t_i}$. Por lo tanto, se puede determinar una estimación del error en el instante t_f ($\widehat{\boldsymbol{\varepsilon}}_{t_f}$) y, finalmente, calcular el valor de $\widehat{\boldsymbol{x}}_{t_f}$ como:

$$\widehat{\boldsymbol{x}}_{t_f} = \boldsymbol{x}^{\mathcal{P}}_{t_f} + \widehat{\boldsymbol{\varepsilon}}_{t_f}. \tag{3-3}$$

En resumen, la metodología híbrida permite combinar un propagador clásico junto con un modelo predictivo que incrementa su precisión sin aumentar significativamente su tiempo de cálculo. Esta técnica no invasiva no requiere modificar el código del propagador. Su impacto en el tiempo de computación depende del número de series temporales $\boldsymbol{\varepsilon}_{t_i}$ modeladas, del conjunto de variables utilizadas y de los métodos predictivos seleccionados.

Es importante destacar que los propagadores numéricos, analíticos o semianalíticos únicamente necesitan información sobre la posición y velocidad en un instante inicial t_0; sin embargo, el propagador híbrido ($\mathcal{HP}$) requiere dos elementos adicionales. En primer lugar, precisa de una secuencia de posiciones y velocidades para entrenar el modelo predictivo. En segundo lugar, necesita una secuencia de datos que permita al modelo predictivo realizar estimaciones desde el instante inicial t_0 en adelante.

3.2. Metodología

Predecir la trayectoria de un satélite artificial es un problema intrínsecamente complejo debido a las perturbaciones que pueden afectar a su órbita alrededor de la Tierra. Algunas de ellas pueden ser modeladas matemáticamente con precisión, mientras que otras contienen incertidumbres que son difíciles de abordar, como se explicó en la sección 1.2. Los efectos que producen las perturbaciones sobre el satélite varían según su altitud respecto al planeta (figura **1-3**). Los propagadores orbitales aprovechan, principalmente, esta característica para incorporar distintos efectos en el modelo de perturbaciones que se evalúa, lo que les permite ser más eficientes desde el punto de vista computacional.

La metodología presentada en este apartado para desarrollar un propagador híbrido puede aplicarse a todo tipo de propagadores, ya sean numéricos, analíticos o semi-analíticos. Esta versatilidad se debe a su naturaleza no invasiva, la cual no requiere modificar el código original del propagador.

La metodología que posibilita la hibridación consta de tres fases fundamentales:

1. Análisis preliminar.
2. Selección de la técnica de predicción.
3. Construcción del propagador híbrido.

Además de la elección del propagador orbital $\mathcal{P}$, la selección del tipo de órbita o la región del espacio $\mathcal{R}$ donde se encuentra el satélite resulta fundamental. Como se mencionó anteriormente, la parte híbrida tiene la tarea de modelar, entre otras cosas, los efectos de las perturbaciones que no son consideradas en el propagador orbital. La altitud del satélite

respecto a la Tierra tiene un papel esencial en la caracterización de las series temporales que se pretende estimar, ya que esta característica influye sobre los efectos que le producen las perturbaciones.

El objetivo principal de esta primera etapa es determinar el número óptimo de series temporales a modelar en la parte híbrida. Manteniendo la premisa de controlar la eficiencia computacional del propagador original, se busca desarrollar propagadores híbridos de forma parsimoniosa. En otras palabras, el propósito es seleccionar el menor número de series temporales que minimice el máximo del error entre la posición real del objeto y la estimada. En la segunda etapa se seleccionan las técnicas de predicción estadísticas o de IA en función de las características de las series temporales a estimar. Por último, en la tercera etapa se ensambla el módulo predictivo con el propagador orbital de partida.

De esta manera, la metodología híbrida convierte el problema dinámico en un problema de predicción de series temporales. Por consiguiente, resulta fundamental elegir una métrica adecuada para evaluar la precisión de las estimaciones proporcionadas por los métodos de predicción. Además, hay que tener en cuenta que en las dos primeras etapas, se dispone de los datos que generaron todas las series temporales, es decir, las efemérides precisas y las proporcionadas por $\mathcal{P}$. Esto no solo permite determinar el error de la estimación en las series temporales, sino también evaluar directamente cómo la estimación de estas series afecta al error en la posición del objeto. Las métricas comúnmente utilizadas en el ámbito de la propagación orbital incluyen el error en distancia, que representa la distancia euclidiana entre la posición real y la predicha por $\mathcal{P}$, así como los errores *along-track*, *cross-track* y *radial* [142, 143, 144]. A continuación, se describen las tres etapas junto con las técnicas y métodos empleados en cada una de ellas.

3.3. Análisis preliminar

En este apartado se analiza el comportamiento del propagador $\mathcal{P}$ en la región $\mathcal{R}$ en función de su precisión. Para ello, se realizará un análisis del error seleccionado entre $\mathcal{P}$ y un conjunto de observaciones precisas $\mathcal{O}$ o, en su defecto, pseudo-observaciones generadas por un propagador orbital de alta precisión en la región $\mathcal{R}$, para m objetos distintos en diferentes épocas, manteniendo un horizonte de propagación fijo. A partir de las seis series temporales $\boldsymbol{\varepsilon}_t$, las cuales contienen los errores entre el propagador y las observaciones precisas, se realizará un Análisis Exploratorio de Datos (EDA). El propósito de este análisis es identificar las series temporales que se incluirán en la parte híbrida. Para ello, se analizará la influencia de cada una de las variables y de todas sus posibles combinaciones en la precisión de $\mathcal{P}$.

En primer lugar, se selecciona el conjunto de variables para representar $\boldsymbol{\varepsilon}_t$ y llevar a cabo el análisis. Es importante destacar que cada conjunto de variables describe el comportamiento del objeto de diferente manera. Entre los conjuntos que se pueden utilizar se incluyen las coordenadas cartesianas, los elementos orbitales (o su versión canónica las variables de Delaunay), las variables equinociales o las polares-nodales, entre otras. Las variables de Delaunay, las polares-nodales y los elementos equinociales se definen en función de los elementos orbitales (ver apartado 1.1.3) mediante las siguientes expresiones:

- Variables de Delaunay:

$$(l = M, g = \omega, h = \Omega, L = \sqrt{\mu a}, G = L\sqrt{1 - e^2}, H = G\cos i).$$

- Variables polares-nodales:

$$\left(r = \frac{a\,(1 - e^2)}{1 + e\cos f}, \theta = \omega + f, \nu = \Omega, R = \frac{e\,G\sin f}{a\,(1 - e^2)}, \Theta = G, N = H\right),$$

donde f representa a la anomalía verdadera. θ es el argumento de la latitud que relaciona el argumento del perigeo y la anomalía media f. ν representa el argumento del nodo. G relaciona el semieje mayor a y la excentricidad e (es decir la forma de la órbita) y se conoce como el módulo del momento angular del satélite. Por otro lado, la variable H depende de a, e e i y define la orientación del plano orbital representando la proyección del vector momento angular del orbitador sobre el eje Z.

- Elementos equinociales:

$$\left(a,\ h = e\sin\zeta,\ k = e\cos\zeta,\ p = \left[\tan\left(\frac{i}{2}\right)\right]^{\mathrm{I}}\sin\Omega,\ \ q = \left[\tan\left(\frac{i}{2}\right)\right]^{\mathrm{I}}\cos\Omega,\ \lambda = M + \zeta\right),$$

donde $\zeta = \omega + \mathrm{I}\Omega$. La cantidad I se denomina factor retrógrado y puede tomar dos posibles valores:

$$\mathrm{I} = \begin{cases} +1 & \text{para los elementos equinociales directos,} \\ -1 & \text{para los elementos equinociales retrógrados.} \end{cases}$$

Los elementos orbitales son el punto de partida, ya que proporcionan una descripción geométrica de las diferencias entre $\mathcal{O}$ y $\mathcal{P}$. A continuación, se extiende este análisis a otros conjuntos de variables para obtener modelos parsimoniosos, es decir, modelos simples con menos suposiciones y variables, pero con una alta capacidad explicativa.

Para cada uno de los m objetos, las efemérides correspondientes de $\mathcal{O}$ y $\mathcal{P}$, normalmente proporcionadas en coordenadas cartesianas, se transforman a elementos orbitales. El análisis se inicia evaluando cada variable por separado. Por ejemplo, para evaluar el impacto de corregir el error del semieje mayor de $\mathcal{P}$, $a^{\mathcal{P}} + \varepsilon^a$ en el error seleccionado, se reemplaza la columna correspondiente en el fichero de $\mathcal{P}$ por la mejor estimación posible del semieje mayor, la cual es la proporcionada por $\mathcal{O}$, $a^{\mathcal{O}}$. Los demás elementos orbitales no se modifican: ($a^{\mathcal{O}}$, $e^{\mathcal{P}}$, $i^{\mathcal{P}}$, $\Omega^{\mathcal{P}}$, $\omega^{\mathcal{P}}$, $M^{\mathcal{P}}$). Posteriormente, una vez realizado este cambio, se calcula el máximo del error seleccionado respecto a $\mathcal{O}$ a lo largo de todo el periodo de propagación. Finalmente, se realiza el mismo proceso para estudiar el error basado en las $2^6 = 64$ posibles combinaciones de variables, incluyendo el caso en que ninguna variable es reemplazada y el que son reemplazadas las seis variables. En este último caso la reducción del error es máxima, lo que conduce a errores nulos.

Análisis Exploratorio de Datos

Una vez determinado el máximo del error seleccionado para cada una de las 64 combinaciones de elementos orbitales en cada uno de los m objetos, se procederá a clasificar estas combinaciones según su capacidad para reducir el error. Para ello, se llevará a cabo:

- Un *Análisis de Componentes Principales*, PCA por sus siglas en inglés. El PCA es una técnica estadística empleada para simplificar la complejidad en conjuntos de datos de alta dimensionalidad, preservando al mismo tiempo información esencial [145]. El objetivo principal del PCA consiste en transformar un conjunto de variables originales en un nuevo conjunto de variables no correlacionadas, conocidas como Componentes Principales (CP en adelante). Estas CP son combinaciones lineales de las variables originales y se ordenan de manera que la primera retiene la mayor variabilidad en los datos, seguida por la segunda, y así sucesivamente.
- Un estudio gráfico de errores utilizando diagramas de caja y bigotes, *boxplot* en inglés, (figura **3-1**). Este tipo de gráfico representa visualmente un conjunto de datos mediante una caja que abarca la mediana, donde los bordes inferior y superior de la caja delimitan el primer y el tercer cuartil (Q_1 y Q_3, respectivamente). Los bigotes, que se extienden desde cada borde de la caja, representan los límites del rango de valores típicos. Los datos fuera de este rango se consideran *valores atípicos* y se representan de forma individual, habitualmente, con puntos circulares. Los límites superior e inferior de este rango se determinan mediante $Q_3 + 1.5\times(Q_3 - Q_1)$ y Q_1 - $1.5\times(Q_3 - Q_1)$, respectivamente.

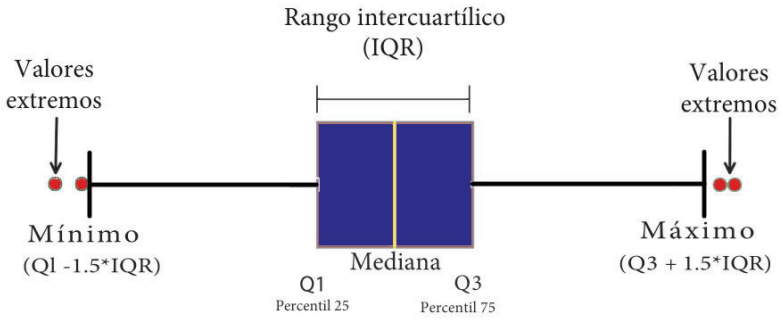


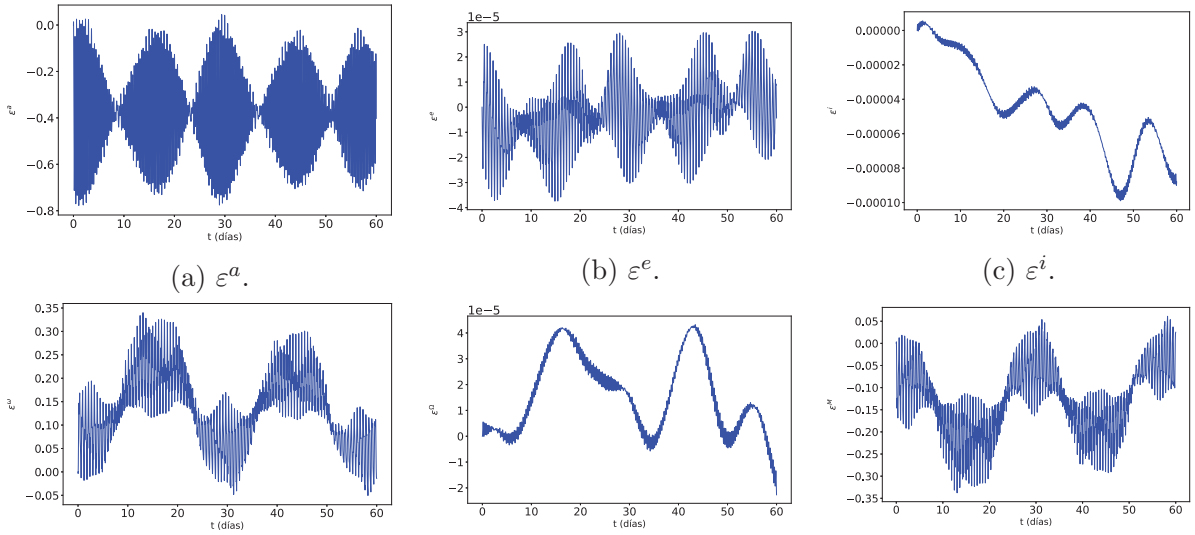
Figura 3-1.: Gráfico de caja y bigotes, nótese que el rango intercuartílico es IQR=$Q_3 - Q_1$.

Este estudio se realiza utilizando diferentes conjuntos de variables con el fin de identificar las series temporales que se incluirán en la parte híbrida.

Comportamiento de las series temporales

El objetivo de este estudio es realizar un análisis en profundidad del comportamiento de las series temporales que tienen un mayor impacto sobre el error. Este análisis es fundamental para seleccionar las técnicas de estimación más adecuadas.

En la figura **3-2** se presenta un ejemplo del comportamiento de las series temporales del error para uno de los satélite de la constelación Galileo. Estas series fueron calculadas utilizando SGP4 y el propagador numérico AIDA durante un período de 60 días. Las gráficas de secuencias muestran las variaciones tanto en las componentes seculares como en las periódicas de largo y corto período que afectan a los elementos orbitales. Como se puede suponer, el comportamiento de los errores dependerá del propagador utilizado, de las observaciones precisas o, en su defecto, de las pseudo-observaciones generadas por un propagador numérico, de la región específica del espacio a la que pertenezca el estado inicial y del tiempo de propagación.



(a) ε^a. (b) ε^e. (c) ε^i.

(d) ε^ω. (e) ε^Ω. (f) ε^M.

Figura 3-2.: Series temporales del error entre SGP4 y un propagador numérico para un satélite de la constelación Galileo.

Para identificar patrones periódicos o cíclicos en las series temporales del error se pueden emplear herramientas como la función de autocorrelación [146], la función de autocorrelación parcial [147] o el periodograma. Por otro lado, para analizar la tendencia se dispone del método estadístico no paramétrico de *Mann-Kendall* [148].

3.3.1. Selección de la técnica de predicción

Una vez que se identifican y caracterizan las series del error, la cantidad y la complejidad de los datos disponibles, y el propósito que se persigue con el propagador híbrido; se elige el método predictivo, ya sea estadístico o basado en técnicas de IA.

Los modelos estadísticos se basan en el supuesto de que los valores pasados pueden utilizarse para predecir valores futuros. Estos modelos son relativamente sencillos de entender y aplicar, especialmente en series temporales donde los datos no presentan relaciones complejas. Algunos de los modelos clásicos más comunes incluyen el modelo auto-regresivo integrado de promedio móvil (en inglés *Autoregressive Integrated Moving Average*, ARIMA) [149] el

modelo de suavizado exponencial [150] y el modelo vector autorregresivo (en inglés *Vector Auto-Regression*, VAR) para el análisis de series temporales multivariadas [151]. Estos modelos tienen la desventaja de ser sensibles a la presencia de valores atípicos y a la no estacionariedad; y, en ocasiones, pueden no ser capaces de captar las tendencias a largo plazo de los datos.

A diferencia de los modelos estadísticos, los modelos basados en IA son capaces de identificar y reproducir patrones complejos, relaciones no lineales y dependencias a largo plazo en los datos de las series temporales. Entre las técnicas basadas en IA destacan las redes neuronales profundas (conocidas en inglés como *Deep Neural Networks*, DNN), las redes Transformer [152], las redes Convolucionales (en inglés *Convolutional Neural Networks*, CNN) [93], y las redes neuronales recurrentes (en inglés *Recurrent Neural Network*, RNN) [92], junto a sus variaciones como las redes formadas por unidades GRU (en inglés *Gated Recurrent Unit*, GRU) [153] y las redes de neuronales memoria de corto-largo plazo (en inglés Long Short-Term Memory, LSTM) [154].

La figura **3-3** proporciona una visión general de la secuencia de operaciones que conduce a la creación del modelo predictivo, utilizando tanto técnicas estadísticas como de IA. En este diagrama, el color rojo representa los procesos relacionados con el modelado de las series temporales, mientras que el proceso de predicción se indica en verde. Además, se describen en azul otros procesos, que van desde el preprocesamiento de los datos hasta la generación de gráficos para monitorear los procesos intermedios y presentar resultados.

3.3.2. Propagador híbrido

El Algoritmo 1 muestra el pseudocódigo de un propagador híbrido basado en el propagador $\mathcal{P}$, que modeliza las series temporales $\boldsymbol{\varepsilon}_{t_f}$ para el estado del objeto $\boldsymbol{x}_{t_0}$. Los datos de entrada son las condiciones iniciales $\boldsymbol{x}_{t_0}$ en el instante inicial t_0 y el instante final t_f para el propagador orbital $\mathcal{P}$, mientras que la parte híbrida requiere los coeficientes del módulo predictivo $\mathcal{C}_{\mathcal{M}}$, los n elementos que inicializan el módulo predictivo por cada serie temporal del error que se desee estimar $(st_{-n+1}, \ldots, st_{-1}, st_0)$ y el instante final t_f. A partir de estos datos iniciales se calcula $\boldsymbol{x}_{t_f}^{\mathcal{P}}$ utilizando $\mathcal{P}$ y $\widehat{\boldsymbol{\varepsilon}}_{t_f}$ con el módulo predictivo $\mathcal{F}$. En caso de requerirse un ajuste en t_f, este módulo debe incluir un sistema de interpolación. Ambos cálculos pueden ser realizados de forma independiente y, por tanto, paralelizable. Finalmente, la estimación del error se añade al estado calculado utilizando el propagador.

Algoritmo 1: Propagador híbrido basado en $\mathcal{P}$

Datos: $\boldsymbol{x}_{t_0}$, $\boldsymbol{st}_{t_0}$, $\mathcal{C}_{\mathcal{M}}$, t_f

Resultado: $\widehat{\boldsymbol{x}}_{t_f}$

$\boldsymbol{x}_{t_f}^{\mathcal{P}} = \mathcal{P}(t_f, \boldsymbol{x}_{t_0})$;

$\widehat{\boldsymbol{\varepsilon}}_{t_f} = \mathcal{F}(\mathcal{C}_{\mathcal{M}}, t_f, \boldsymbol{st}_{t_0})$;

$\widehat{\boldsymbol{x}}_{t_f} = \boldsymbol{x}_{t_f}^{\mathcal{P}} + \widehat{\boldsymbol{\varepsilon}}_{t_f}$;

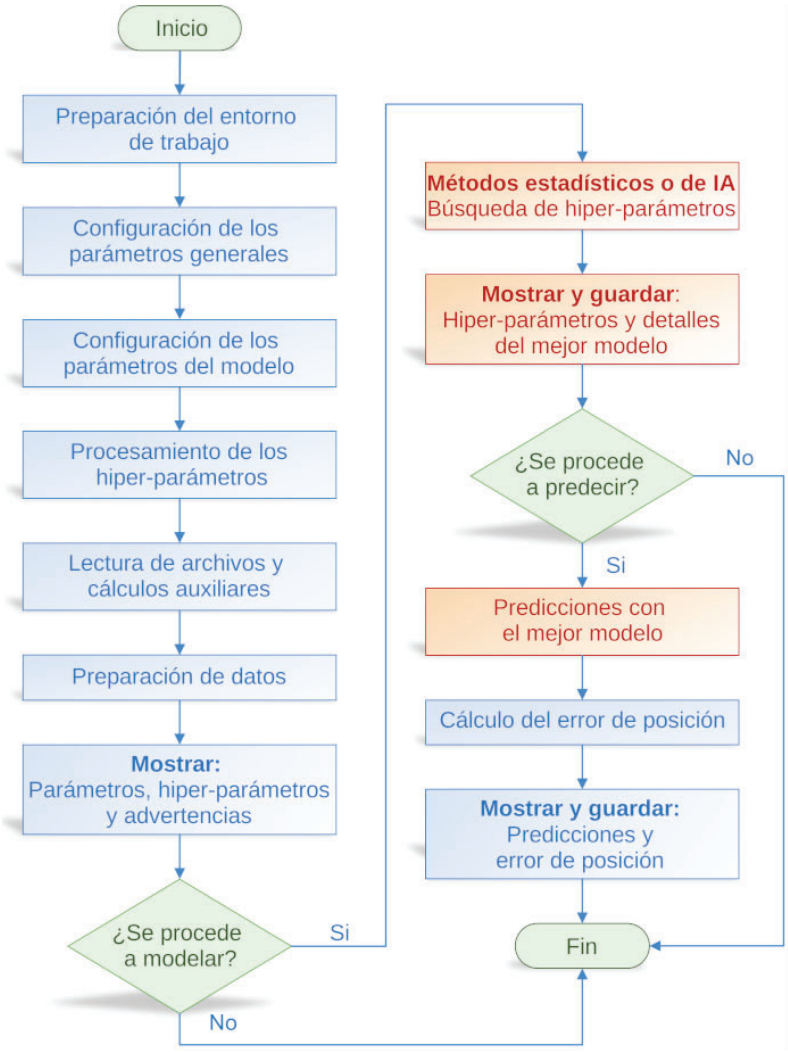


Figura 3-3.: Diagrama de flujo que muestra la secuencia de operaciones de modelización y predicción utilizando técnicas estadísticas y de IA.

Es importante destacar que, para cada condición inicial $\boldsymbol{x}_{t_0}$, es necesario calcular el conjunto correspondiente de coeficientes $\mathcal{C}_{\mathcal{M}}$ para el módulo predictivo.

Si la técnica de predicción tiene la capacidad de modelar más de un patrón, se observa que el conjunto de coeficientes $\mathcal{C}_{\mathcal{M}}$ se comparte entre un grupo de condiciones iniciales. Esto conduce a la creación del Algoritmo 2, que representa una versión híbrida del método de Encke basada en el propagador $\mathcal{P}$. En este escenario el módulo predictivo desempeña el papel de modelar las diferencias entre $\mathcal{P}$ y las observaciones precisas; de manera análoga a la integración de las diferencias entre el intermediario y el modelo preciso de perturbaciones, como se observa en la ecuación (1-34) del método de Encke.

Por último, se van a introducir dos índices que permiten evaluar el desempeño de los propagadores híbridos. El primero mide la mejora del propagador híbrido respecto del propagador hibridado $\mathcal{P}$. El segundo mide la capacidad de mejora del propagador híbrido respecto a

Algoritmo 2: Propagador Encke-híbrido basado en $\mathcal{P}$ válido en $\mathcal{R}$

Datos: $\boldsymbol{x}_{t_0}$, $\boldsymbol{st}_{t_0}$, t_f

Resultado: $\widehat{\boldsymbol{x}}_{t_f}$

$\boldsymbol{x}^{\mathcal{P}}_{t_f} = \mathcal{P}(t_f, \boldsymbol{x}_{t_0})$;

$\widehat{\boldsymbol{\varepsilon}}_{t_f} = \mathcal{F}(t_f, \boldsymbol{st}_{t_0})$;

$\widehat{\boldsymbol{x}}_{t_f} = \boldsymbol{x}^{\mathcal{P}}_{t_f} + \widehat{\boldsymbol{\varepsilon}}_{t_f}$;

un propagador híbrido ideal u óptimo, $\mathcal{HP}_{\text{Opt}}$, que representa al propagador híbrido que minimiza el error seleccionado.

1. El índice de mejora respecto a $\mathcal{P}$

   Sea error$(\mathcal{P}) \neq 0$, se define el *índice de mejora* respecto a $\mathcal{P}$, $\mathcal{I}_{m_\mathcal{P}}$, como el cociente entre el error cometido por un propagador híbrido ($\mathcal{HP}$) respecto del error cometido por el propagador hibridado ($\mathcal{P}$).

   $$\mathcal{I}_{m_\mathcal{P}} = \frac{\text{error}(\mathcal{HP})}{\text{error}(\mathcal{P})}, \tag{3-4}$$

   donde error() es una medida que permita evaluar la precisión de un propagador orbital. El índice de mejora puede tomar valores entre $0 < \mathcal{I}_{m_\mathcal{P}} < \infty$.

2. El índice de mejora respecto a $\mathcal{HP}_{\text{Opt}}$

   Sea error$(\mathcal{HP}) \neq 0$, se define el *índice de mejora* respecto a $\mathcal{HP}_{\text{Opt}}$, $\mathcal{I}_{m_{\mathcal{HP}_{\text{Opt}}}}$, como el cociente entre el error cometido por el propagador hibrido ideal ($\mathcal{HP}_{\text{Opt}}$) respecto del error cometido por un propagador híbrido ($\mathcal{HP}$).

   $$\mathcal{I}_{m_{\mathcal{HP}_{\text{Opt}}}} = \frac{\text{error}(\mathcal{HP}_{\text{Opt}})}{\text{error}(\mathcal{HP})}, \tag{3-5}$$

   donde error() es una medida que permita evaluar la precisión de un propagador orbital. El índice de mejora puede tomar valores entre $0 < \mathcal{I}_{m_{\mathcal{HP}_{\text{Opt}}}} \leq 1$.

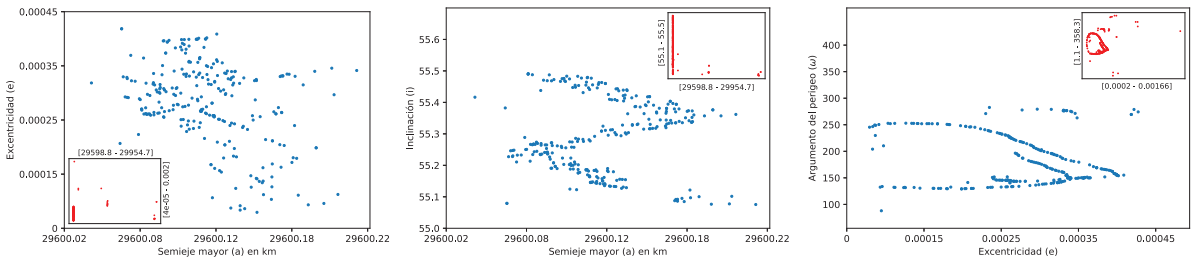
3.4. Análisis preliminar aplicado a SGP4

En esta sección se analiza el comportamiento del propagador SGP4 en la región MEO en función de su precisión, con el objetivo de generar un propagador orbital de tipo Encke, $\text{HEncke}_{\text{SGP4}}$, que utilice a SGP4 como intermediario. Para realizar este análisis, se evaluará el error en distancia entre SGP4 y las pseudo-observaciones generadas por el propagador numérico de alta precisión AIDA, descrito en la sección 1.3.1, utilizando un conjunto de TLE de la constelación Galileo. El propósito es identificar y caracterizar las variables cuyos errores serán estimados en la parte híbrida del propagador de tipo Encke.

3.4.1. Datos de estudio

En este trabajo se utilizarán dos conjuntos de TLE, obtenidos de Space-Track.org[1], que representan las condiciones iniciales de dos satélites pertenecientes a la constelación Galileo, el sistema de posicionamiento global europeo que se encuentra en la región MEO. Concretamente, son 313 TLE del satélite GSAT0203 y 1688 TLE del satélite GALILEO–FM3. La constelación Galileo se encuentra en la región MEO. Sus identificadores NORAD son 40545 y 38857, respectivamente. El primero fue lanzado el 27 de marzo de 2015; mientras que el segundo, uno de los cuatro satélites iniciales de la constelación, fue lanzado el 12 de octubre de 2012. Se utilizarán los TLE del satélite GSAT0203 para determinar las mejores arquitecturas que se implementarán en la parte híbrida del propagador de tipo Encke, mientras que los TLE correspondientes a GALILEO–FM3 serán empleados para validar los modelos entrenados.

En la figura **3-4** se presenta las relaciones entre los pares de elementos orbitales (a, e), (a, i) y (e, ω) de los TLE del satélite GSAT0203. Estos TLE abarcan el período desde el 19 de abril de 2015 hasta el 28 de diciembre de 2016.



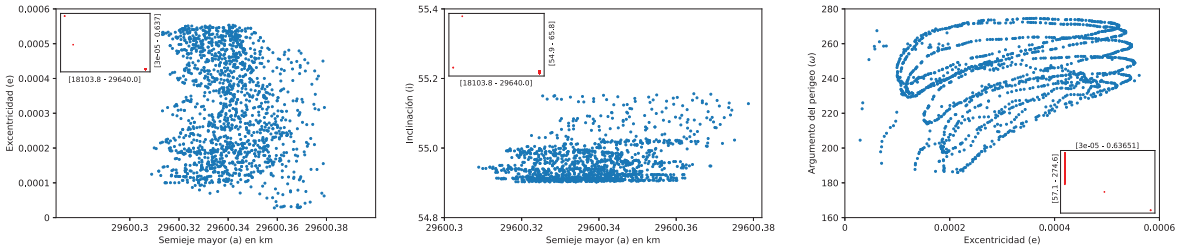
(a) a vs e (b) a vs i (c) e vs ω

Figura 3-4.: Distribución de los TLE del satélite GSAT0203. Las ventanas dentro de los gráficos representan el mismo diagrama de dispersión, pero en rangos más amplios de las mismas variables. De este modo, la ventana grande muestra la región más densa.

En la figura **3-4**a, se observa que los valores de a se encuentran entre 29598.8 y 29954.7 km, pero el 90.76 % de los casos están en el rango de 29600.02 a 29600.22 km. Los valores de e varían entre 0.00004 y 0.002, con una concentración del 90.45 %, en el intervalo de 0.00004 a 0.00045. En cuanto a los valores de i, estos están entre 55° y 55.6°, como se muestra en la figura **3-4**b. Los valores de ω se encuentran en el rango de 1.1519° a 358.291°, con una mayor concentración, alrededor del 98.73 %, en el intervalo de 120° a 260° como se observa en la figura **3-4**c. Finalmente, el argumento del nodo varía entre 76.8346° y 94.9086°, mientras que la anomalía media varía entre 1.685° y 358.952°.

En la figura **3-5**, se describe las relaciones entre los pares de elementos orbitales (a, e), (a, i) y (e, ω) de los TLE para el caso del satélite GALILEO–FM3. Estos TLE comprenden el período desde el 12 de octubre de 2012 hasta el 13 de marzo de 2021.

[1] `https://www.space-track.org`.



(a) a vs e. (b) a vs i (c) e vs ω

Figura 3-5.: Distribución de los TLE del satélite GALILEO-FM3.

En este conjunto de TLE, los valores de a se encuentran en el rango de 18103.8 a 29640.0 km, pero el 94.43 % se encuentra entre 29600.30 y 29600.38 km. Las excentricidades están en el intervalo de 0.00003 a 0.637, con un 99.76 % de los casos en el rango de 0.00003 a 0.00006. En cuanto a los valores de i, estos se sitúan entre 54.9° y 65.8°, con un 89.63 % entre 54.8° y 55.2°. Los valores de ω están en el rango de 57.1° a 274.6°, mostrando su mayor concentración del 94.96 % en el intervalo entre 180° y 280°.

3.4.2. Análisis exploratorio de datos

En este apartado se analiza el impacto que puede tener la corrección del error de cada una de las variables, o de sus combinaciones sobre la precisión del propagador SGP4, a partir de pseudo-observaciones generadas por el propagador numérico de alta precisión AIDA.

Cada uno de los 313 TLE del satélite GSAT0203 se propaga con SGP4 durante un periodo de 30 días, generando una salida a intervalos de un segundo. Las efemérides obtenidas son coordenadas cartesianas referidas al sistema de referencia TEME (acrónimo en inglés de True Equator Mean Equinox frame). Posteriormente, estas coordenadas se transforman al sistema de referencia J2000 y se almacenan en un fichero de texto. Por otro lado, se toman las coordenadas cartesianas de SGP4 en el instante inicial referidas a J2000 y se propagan con AIDA durante 30 días, generando también una salida a intervalos de un segundo. AIDA produce directamente coordenadas cartesianas referidas al sistema J2000, las cuales se almacenan en un fichero de texto. El proceso de propagación concluye con el cálculo de los elementos orbitales y de las variables de Hill a partir de los archivos de coordenadas cartesianas. Es importante destacar que los errores en distancia entre AIDA y SGP4 varían en un rango desde aproximadamente 1.31 km (mínimo) hasta 52.01 km (máximo), con una mediana de 10.73 km.

A continuación, se presenta parte de este análisis exploratorio de datos utilizando los elementos orbitales y las variables polares-nodales. Para más información ver [155, 156].

En el caso de los elementos orbitales hay una mejora en el error en distancia del propagador SGP4 con respecto a AIDA en solo 16 de las 64 posibles combinaciones para todos los TLE.

Estas combinaciones son:

$$\{(a,e,i,\Omega,\omega,M),(a,e,i,\omega,M),(a,e,\Omega,\omega,M),(a,i,\Omega,\omega,M),(e,i,\Omega,\omega,M),$$
$$(a,e,\omega,M),(a,i,\omega,M),(a,\Omega,\omega,M),(e,i,\omega,M),(e,\Omega,\omega,M),(i,\Omega,\omega,M),$$
$$(e,\omega,M),(a,\omega,M),(i,\omega,M),(\Omega,\omega,M),(\omega,M)\},$$

como se puede observar, todas las combinaciones incluyen los argumentos del perigeo y del nodo, siendo por tanto, las variables que más influencia tienen en la disminución del error en distancia.

En la figura **3-6** se muestran los gráficos de caja y bigotes que incluyen las combinaciones de elementos orbitales con los menores errores en distancia entre AIDA y SGP4 que contienen menos de cinco variables. En el caso de la combinación (ω, M), los errores en distancia varían desde aproximadamente 1.2 km (mínimo) hasta 3.6 km (máximo), con una mediana de 2.1 km. Es importante destacar que estos valores marcan la máxima reducción del error que podría lograr un propagador híbrido que estimara solo estas dos variables.

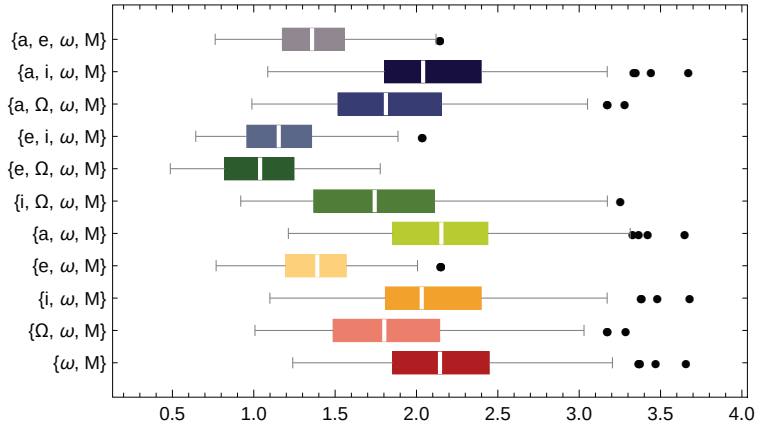


Figura 3-6.: Errores en distancia (km) correspondientes a las mejores combinaciones en elementos orbitales durante un periodo de 30 días.

Sin embargo, en el caso de las variables polares-nodales el número de combinaciones que reducen el error en distancia asciende a 32. Estas combinaciones son:

$$\{(r,\theta,\nu,R,\Theta,N),(\theta,\nu,R,\Theta,N),(r,\theta,R,\Theta,N),(r,\theta,\nu,\Theta,N),(r,\theta,\nu,R,N),$$
$$(r,\theta,\nu,R,\Theta),(\theta,R,\Theta,N),(\theta,\nu,\Theta,N),(\theta,\nu,R,N),(\theta,\nu,R,\Theta),(r,\theta,\Theta,N),$$
$$(r,\theta,R,N),(r,\theta,R,\Theta),(r,\theta,\nu,N),(r,\theta,\nu,\Theta),(r,\theta,\nu,R),(\theta,\Theta,N),(\theta,R,N),$$
$$(\theta,R,\Theta),(\theta,\nu,N),(\theta,\nu,\Theta),(\theta,\nu,R),(r,\theta,N),(r,\theta,\Theta),(r,\theta,R),(r,\theta,\nu),$$
$$(\theta,N),(\theta,\Theta),(\theta,R),(\theta,\nu),(r,\theta),(\theta)\}.$$

En este caso, todas las combinaciones incluyen el argumento de la latitud θ. Es importante recordar que $\theta = \omega + f$ y que la anomalía verdadera f está relacionada con M a través de la ecuación de Kepler.

En la figura **3-7**, se muestran los gráficos de caja y bigotes con las combinaciones de tres o menos variables polar-nodales que reducen el error en distancia entre AIDA y SGP4. En el

caso de la corrección única en el argumento de la latitud (θ), los errores en distancia varían desde aproximadamente 1 km (mínimo) hasta 2.4 km (máximo), con una mediana de 1.5 km. Estos resultados son ligeramente mejores que los obtenidos por la combinación (ω, M) en elementos orbitales.

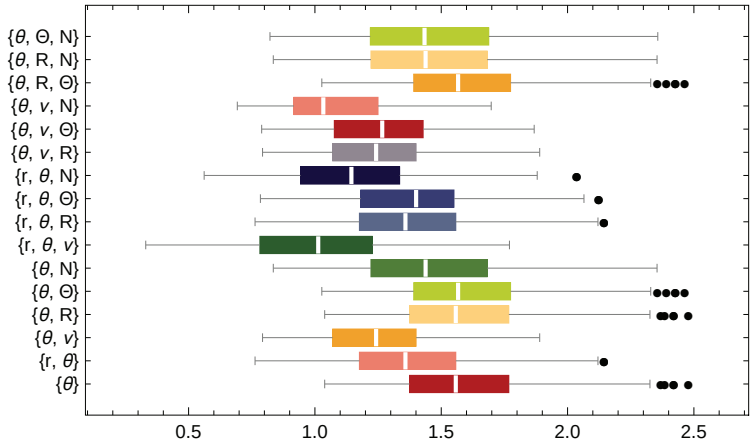


Figura 3-7.: Errores en distancia (km) correspondientes a las mejores combinaciones en variables polares-nodales durante un periodo de 30 días.

Por tanto, estimando únicamente el argumento de la latitud se consigue una mejora notable en comparación con SGP4. Esta mejora se extiende a todos los TLE del satélite GSAT0203, reduciendo en todos los casos los errores en distancia entre AIDA y SGP4. Para este trabajo se considerará un modelo híbrido de tipo Encke basado en SGP4 que exclusivamente modelizará la variable θ en la parte híbrida. Es importante destacar que los resultados obtenidos durante este estudio nos proporcionan la mejor estimación del error que se puede obtener con cualquier modelo predictivo que solo incluya el error en el argumento de latitud.

3.4.3. Series del error en el argumento de latitud

En este apartado se examina el comportamiento de las 313 series temporales del error en la variable argumento de la latitud, $\varepsilon_i^\theta = \theta_i^{AIDA} - \theta_i^{SGP4}$, que corresponden a los TLE de los satélites GSAT0203 y GALILEO–FM3 durante 30 días de propagación. Desde el punto de vista geométrico, $\epsilon_i^\theta > 0$ indica que la posición de AIDA está adelantada con respecto a la de SGP4, mientras que $\epsilon_i^\theta < 0$ indica que está retrasada; además, la magnitud refleja la distancia entre ambas posiciones.

La gráfica de secuencias con las 313 series temporales ε_i^θ se muestra en la figura **3-8**. En primer lugar, se observa un comportamiento secular o tendencia, caracterizado principalmente por una progresión lineal, tanto positiva como negativa, en el nivel de estas series. En segundo lugar, se identifica un patrón estacional o periódico en el que se observan oscilaciones con aparente regularidad.

Para determinar la frecuencia de repetición de estas oscilaciones se emplea la función de autocorrelación y el periodograma. Ambas técnicas ponen de manifiesto que estas oscilaciones se repiten, aproximadamente, cada 14.28, 28.56, 42.84 horas, y así sucesivamente. Estos

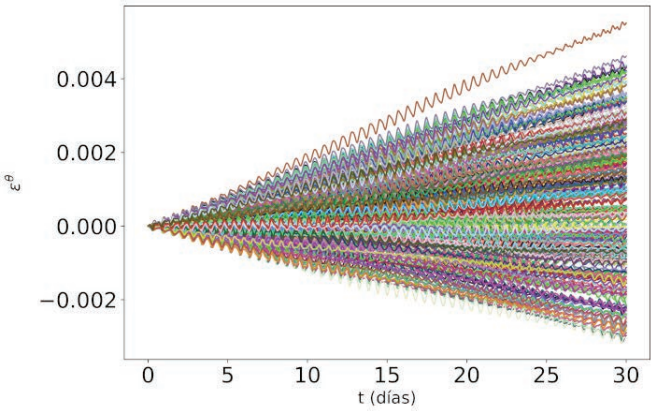


Figura 3-8.: Gráficas de secuencia de las 313 series temporales ε^θ.

valores coinciden, aproximadamente, con múltiplos del periodo kepleriano, ecuación (1-19), de los TLE de GSAT0203, los cuales se encuentran en un rango de entre 14.07 y 14.33 horas.

Sin embargo, la tendencia es la característica que permite clasificar las series temporales en tres grupos: los dos primeros muestran una tendencia que refleja un crecimiento o decrecimiento a lo largo del tiempo, mientras que en el tercer grupo no se aprecia esta componente. En este análisis se utilizó la técnica estadística no paramétrica de Mann-Kendall y la de agrupamiento K-means. Primero, se estimó la pendiente de cada una de las series y se ordenaron los valores de menor a mayor. A continuación, se empleó la técnica de agrupamiento K-means con $k = 3$ para dividir las series en grupos. Como resultado, se obtuvieron 169, 118 y 26 series temporales con tendencias positivas, negativas y sin tendencias, respectivamente.

En el caso de las series temporales correspondientes al satélite GALILEO–FM3 se observa un comportamiento similar al observado para las series de GSAT0203. En este caso, el grupo mayoritario contiene 1444 series con tendencia positiva, seguido por el grupo de 226 series con tendencia negativa y el grupo de 18 series sin tendencia.

Es importante destacar que se suprimirá de este trabajo una de las series temporales en el caso de GSAT0203 y tres en el caso de GALILEO–FM3 debido a que sus TLE no se encuentran exactamente en la región MEO. Posiblemente corresponden a épocas en las que los satélites estaban realizando una maniobra o, simplemente, son valores atípicos.

En resumen, este análisis demuestra que al reducir únicamente el error en el argumento de la latitud ε^θ, es posible mejorar la precisión de SGP4. Esta premisa es la base sobre la cual construimos un propagador híbrido de tipo Encke basado en SGP4.

4. Propagador híbrido de tipo Encke basado en SGP4: HEncke$_{\mathrm{SGP4}}$

Una vez identificadas las variables cuyos errores se pretenden estimar, se procederá a seleccionar las técnicas de IA más adecuadas. Esta elección se basará en el análisis de las características de las series temporales del error realizado en la primera etapa de la metodología híbrida. En este capítulo se describirá el desarrollo de la segunda y tercera etapa de esta metodología, la cual es aplicada a la construcción de un propagador híbrido de tipo Encke basado en SGP4.

4.1. Introducción

En este trabajo se emplearán redes neuronales (RN) como técnica de predicción para modelar el comportamiento de múltiples series temporales. El modelo de RN se desarrollará en dos iteraciones. En la primera se definirá una arquitectura básica y se evaluarán dos estrategias de entrenamiento para la RN. En la segunda iteración, se optimizará la arquitectura de la RN.

Para cada una de las arquitecturas definidas en cada una de las iteraciones, se entrenarán cinco modelos de RN empleando las series temporales ε^{θ} del satélite GSAT0203. Posteriormente, se validará el mejor modelo con datos del satélite GALILEO–FM3.

La predicción de ε^{θ} se realiza de manera iterativa, tal como se describe en [157]. Este enfoque iterativo en la predicción se basa en el mismo principio que la ventana deslizante. A medida que avanza la predicción la ventana de observaciones se desplaza incorporando estimaciones pasadas y excluyendo valores observados. Este método presenta la ventaja de ser más realista, permitiendo evaluar la capacidad del modelo para mantener la precisión a lo largo del tiempo. Sin embargo, tiene la desventaja de que los errores se acumulan y propagan a lo largo de la predicción.

La evaluación de la capacidad predictiva de los modelos se realizará comparando los pronósticos generados por dichos modelos con los datos del conjunto de prueba para distintos horizontes temporales. En lugar de utilizar las métricas convencionales de series temporales, se simulará el comportamiento del propagador híbrido HEncke$_{\mathrm{SGP4}}$. Estos cálculos seguirán los mismos pasos que los realizados en la primera etapa de esta metodología. En dicha etapa, a partir de los ficheros de efemérides generados por SGP4 y AIDA, se analizó la influencia de cada variable o de sus combinaciones sobre el error en distancia. Esto implica que, para cada serie, se añade la predicción del error del argumento de la latitud $\widehat{\varepsilon}^{\theta}$ a θ^{SGP4} y se calcula el error en distancia respecto a AIDA. Es importante recordar que, en esta etapa, se verificó que la mejor estimación del error se produce cuando el argumento de la latitud de SGP4 es

reemplazado por el de AIDA. Este hecho permite establecer una cota teórica o valor óptimo del error en distancia y un índice que facilita la evaluación de la mejora de la predicción $\widehat{\varepsilon}^{\theta}$ al comparar el error en distancia introducido por esta predicción respecto al valor óptimo.

Para la selección de las arquitecturas y la creación de los modelos de redes neuronales se utilizará un entorno de trabajo cuyas principales funcionalidades se encuentran descritas en la figura **3-3** y documentado en [155].

Este entorno fue inicialmente implementado en R [158], utilizando varios paquetes para la predicción de series temporales estadísticas y haciendo uso de la librería H2O [159, 160] para los métodos de IA. Posteriormente, se ha llevado a cabo una reimplementación y extensión de las funcionalidades de este entorno en Python [161] como parte del trabajo realizado en esta tesis. En esta aplicación, para la preparación de los datos, se hace uso de las librerías de propósito general *statsmodels* [162], *numpy* [163] y *pandas* [164], mientras que para la generación de gráficos de las series se utiliza *matplotlib*, [165] y *seaborn* [166]. En cuanto a las tareas de aprendizaje automático, se integraron dos librerías específicas de ML: *Keras* [167] y *Tensorflow* [168].

Una característica destacable de esta nueva implementación es la generación automática de un programa en C++. Este programa evalúa de manera eficiente el modelo predictivo utilizando la librería Eigen [169], además, se encarga de la parte híbrida del propagador. En el Anexo A se proporciona una descripción más detallada de esta implementación. Ambos entornos son multiplataforma y además están instalados en el sistema de computación de altas prestaciones Beronia de la Universidad de La Rioja. Este clúster de computación cuenta con 25 nodos Hewlett-Packard ProLiant BL460c Gen 9 que representan un total de 764 CPU.

4.2. Primera iteración: arquitectura básica

En esta sección se describe la arquitectura básica de la RN diseñada para modelar simultáneamente el comportamiento de múltiples series temporales. Además, se evalúan dos modos de organizar el conjunto de datos que se utilizarán para entrenar la RN: el modo *secuencial* y el *aleatorio*. En el modo secuencial los datos se organizan de forma ordenada; es decir, se tratan todos los datos de una serie antes de pasar a los datos de la siguiente serie temporal. Por otro lado, en el modo aleatorio los datos de todas las series se mezclan de forma aleatoria antes de ser suministrados a la RN. Para concluir, se procederá a entrenar cinco modelos, todos con la misma arquitectura, utilizando el entorno de trabajo previamente descrito para cada uno de estos modos de organización de datos.

4.2.1. Inicialización del entorno

Para ejecutar el entorno es necesario crear un fichero de configuración que contiene dos tipos de parámetros: generales y específicos. Los parámetros generales abarcan aspectos como el tamaño y la apariencia de las figuras, las épocas para determinar los errores y los nombres de los archivos, entre otros. Por otro lado, entre los parámetros específicos destacan la selección del conjunto de variables a utilizar, el número de valores pasados para la predicción, la longitud de los conjuntos de datos, la resolución de los datos y los

hiperparámetros para el método de predicción.

La determinación de los parámetros específicos de una RN es, frecuentemente, una tarea compleja y computacionalmente costosa que en numerosas ocasiones, requiere el uso de la técnica de prueba y error. En la mayoría de los casos los valores de los hiperparámetros se mantienen en sus configuraciones por defecto ya que, con estos valores, las arquitecturas más básicas se ajustan adecuadamente.

En primer lugar, es importante seleccionar el tamaño de los conjuntos de entrenamiento, validación y prueba. Los dos primeros conjuntos equivalen al intervalo de control de la metodología híbrida. En [170, 155] se examinaron en profundidad la duración del intervalo de control. Ambos estudios concluyeron que la unidad principal para determinar el tamaño de los tres conjuntos es el periodo kepleriano del RSO, es decir, el tiempo que tarda el RSO en completar una revolución. Este periodo marca el límite entre los efectos estacionales de corto y largo periodo. Como se puede observar en el apartado 1.2, el periodo kepleriano coincide, aproximadamente, con el periodo de la componente estacional de las series temporales ε^{θ}.

Además, en [170] se confirmó que la elección de 10 revoluciones y 10 puntos equidistantes por revolución, aproximadamente un punto cada 10 minutos, utilizando técnicas estadísticas para un RSO en una órbita LEO resultó apropiada para el intervalo considerado. Este resultado se validó en [155], en el cual se utilizaron también 10 revoluciones asignando 7 al conjunto de entrenamiento y 3 al de validación, aplicando diversas técnicas de ML en un RSO de la región MEO. El número de puntos por revolución considerados fue 84, equivalente a uno cada 10 minutos. Asimismo, el tamaño del vector de entrada utilizando el método de la ventana deslizante, introducido en la sección 2.7.1, fue de 169 puntos, correspondientes a 2 revoluciones del RSO y un punto extra.

4.2.2. Arquitectura inicial de RN

Una vez fijados los parámetros generales de configuración del entorno, se procede a seleccionar los parámetros específicos de la arquitectura de la RN. Este proceso se describe brevemente en este apartado.

Los valores iniciales de los hiperparámetros se basan en los resultados obtenidos en [155, 156]. En ambos estudios las series temporales se clasificaron según su tendencia como positivas, negativas o sin definir. Se llevó a cabo una búsqueda sistemática de hiperparámetros con el fin de obtener una arquitectura de red óptima para cada uno de los tipos de tendencia. La tabla **4-1** muestra los hiperparámetros de las tres arquitecturas encontradas en [156].

Tendencia	1° capa oculta	2° capa oculta	N° neuronas	Tamaño de bloque	Optimizador
Positiva	linear	elu	256	64	adam
Negativa	tanh	tanh	256	128	adam
Sin definir	relu	tanh	256	64	adam

Tabla 4-1.: Hiperparámetros de las tres arquitecturas de RN.

En la primera columna se especifica la tendencia de la serie. Las siguientes dos columnas

indican, en ese orden, la función de activación utilizada en la primera y la segunda capa oculta. La cuarta columna indica el numero de neuronas de la primera capa oculta. De acuerdo al criterio seguido en [156], el número de neuronas de la segunda capa oculta se reduce a la mitad, por lo que en los tres casos es 128. El tamaño de bloque, o *batch size* por su denominación en inglés, se refiere al número de ejemplos de entrenamiento que se utilizan en una iteración o paso de entrenamiento. La última columna indica el optimizador, el cual coincide en los tres casos.

Como resultado de este análisis, se decide que inicialmente la arquitectura contará con 256 neuronas en la primera capa oculta y se utilizará el optimizador Adam. No obstante, aún falta por determinar la mejor combinación para el tamaño del bloque y las funciones de activación. Los tamaños de bloque pueden ser 64, 128, 256 y 512, mientras que las funciones de activación para cada una de las dos capas ocultas pueden ser *linear*, *tanh*, *elu* y *relu*. La selección óptima de estos hiperparámetros requiere evaluar 64 arquitecturas, las cuales deberán entrenarse simultáneamente con las 312 series temporales. Para reducir el coste computacional de este proceso, se opta por seleccionar aleatoriamente 6 series con tendencia positiva, 6 con tendencia negativa y 6 sin tendencia. Esto reduce el conjunto de entrenamiento a 18 series a partir de las cuales se construyen los conjuntos de entrenamiento y validación.

El conjunto de entrenamiento recoge las 14 primeras revoluciones de las 18 series seleccionadas con la misma tendencia, mientras que las 3 revoluciones siguientes forman el conjunto de validación. Estos conjuntos de datos se descomponen en vectores o muestras mediante el método de la ventana deslizante. En este caso, los vectores correspondientes a una serie aparecen de manera consecutiva tanto en el conjunto de entrenamiento como en el de validación. En el caso del conjunto de entrenamiento se generan 71,772 vectores, cada uno de dimensión 169; mientras que en el conjunto de validación se obtienen 15,456 vectores de la misma dimensión. Cada una de las arquitecturas se entrenó con cinco modelos. El tiempo de cómputo promedio de los modelos fue 2.45 horas. Se dedicaron las siguientes 14 revoluciones para medir la capacidad de predicción de los modelos y su reflejo en la reducción del error en distancia entre los datos de SGP4 y AIDA. En experimentos previos se entrenaron modelos con una única serie. En estos casos se generaron 11,962 vectores de entrenamiento y 2,576 vectores de validación. El tiempo de cómputo para el entrenamiento de estos modelos fue de aproximadamente 0.32 horas.

Por otro lado, se consideró que la tasa de aprendizaje (conocido por el acrónimo LR del inglés *learning rate*) podía variar entre 10^{-4} y 10^{-6}. Con un LR de 10^{-4}, ninguno de los modelos se ajustó a los datos de entrenamiento. En cambio, se observó una mejora significativa al utilizar un LR de 10^{-5}. Aunque con un valor de LR de 10^{-6} se logra una mejora aproximada de 300 m en el error en distancia a los 12 días en comparación con el LR de 10^{-5}, se decide descartar este valor debido a que el tiempo de cómputo adicional no justifica el aumento en precisión obtenido.

Los valores de los hiperparámetros de la mejor arquitectura se muestran en la tabla **4-2**. Es importante destacar que la *paciencia* controla cuánto tiempo se permite que el modelo no mejore antes de detener el entrenamiento, mientras que el *factor de cambio* controla cómo disminuye el LR. En este caso, se consideró reducir el LR multiplicándolo por 0.5 cada vez que pasan 45 épocas sin conseguir una mejora en el ajuste del modelo.

Hiperparámetros	Valores
Número de capas ocultas	2
Nº neuronas en cada capa oculta	256, 128
Funciones de activación en las capas ocultas	tanh, relu
Bloque de entrada	128
Optimizador	adam
Nº épocas de entrenamiento	6000
Paciencia	100
LR	10^{-5}
Paciencia LR	45
Factor de cambio del LR	0.5

Tabla 4-2.: Valores de los hiperparámetros de la arquitectura básica.

4.2.3. Modelo secuencial

En esta etapa se entrenan cinco modelos de RN con la arquitectura definida en la tabla **4-2**. Los conjuntos de entrenamiento, validación y prueba se van a construir seleccionando aleatoriamente 219 series de las 312 disponibles. Cada serie se divide en tres partes. Las primeras 14 revoluciones se emplean en la fase de entrenamiento, las 3 siguientes en la de validación y las restantes 21 en la de prueba. A continuación, los datos de entrenamiento y validación de cada una de las 219 series se combinan en dos conjuntos separados. Estos conjuntos de datos se descomponen en vectores o muestras mediante el método de la ventana deslizante. Es importante destacar que los vectores correspondientes a una serie aparecen de manera consecutiva en los conjuntos de entrenamiento y validación. En el caso del conjunto de entrenamiento, se generan 3,811,401 vectores, cada uno de dimensión 169; mientras que en el conjunto de validación se obtienen 820,999 vectores de la misma dimensión.

El primer paso consistió en clasificar las 219 series de entrenamiento como positivas, negativas o sin tendencia, según los valores de la pendiente mostrados en la tabla **4-3**. La tendencia de las series se calculó utilizando el método estadístico no paramétrico de Mann-Kendall durante los primeros 12 días de propagación. Según este método, el número de series positivas, negativas o sin tendencia es 121, 80 y 18, respectivamente.

Tendencia	Min. pendiente	Max. pendiente
Negativa	-1.64×10^{-06}	-1.02×10^{-07}
Positiva	1.11×10^{-07}	2.55×10^{-06}
Sin tendencia	-9.11×10^{-08}	8.78×10^{-08}

Tabla 4-3.: Clasificación de las series temporales según el estimador de Mann-Kendall durante un periodo de 12 días.

Las 93 series restantes se reservarán para probar la robustez de los modelos de RN. Aunque no ha sido posible que los modelos aprendan su comportamiento específico, es razonable esperar que sea similar al de las 219 series, dado que todas pertenecen al mismo satélite. Además, para evaluar la capacidad de generalización de estos modelos se emplearán las 1,685

series generadas a partir de los datos del satélite GALILEO–FM3. Aunque estas series también muestran comportamientos similares a las anteriores, los patrones que las caracterizan tampoco han sido aprendidos durante el proceso de entrenamiento de los modelos.

El tiempo de cómputo empleado por el clúster Beronia para entrenar los cinco modelos se muestra en la tabla **4-4**. Como se puede observar, el tiempo de los modelos 4 y 5 es significativamente inferior al de los tres primeros. Esto se debe a la activación del hiperparámetro *paciencia*. Según se describe en la sección 2.6.1, la paciencia es uno de los parámetros utilizados en el método de detención temprana (*EarlyStopping*) para monitorizar el rendimiento del modelo y determinar cuándo detener el aprendizaje de manera anticipada. En este caso, el entrenamiento de los modelos 4 y 5 se detuvo anticipadamente después de alcanzar las 100 épocas permitidas por este hiperparámetro sin mostrar una mejora. Es relevante mencionar que esto ha ocurrido a pesar de que la parametrización de la *paciencia LR* y del *factor de cambio del LR*, 45 y 0.5 respectivamente, conlleva que durante 100 épocas la tasa de aprendizaje se reduce a la mitad al alcanzar las 45 épocas y de nuevo al alcanzar las 90.

Modelo	Tiempo (días)
1	24.30
2	24.16
3	28.08
4	8.93
5	11.20

Tabla 4-4.: Tiempo de cómputo de los cinco modelos de RN.

Como se mencionó anteriormente, la capacidad predictiva no se evalúa mediante métricas de series temporales. En su lugar, se simulan los errores en distancia entre AIDA y los propagadores híbridos de tipo Encke que incorporan los modelos de RN, así como con la simulación del propagador óptimo. En esta simulación se reemplaza el argumento de la latitud de SGP4 por el proporcionado por AIDA, lo que permite obtener la mejor estimación del error que se puede alcanzar con un propagador híbrido que solo modeliza esta variable. A partir de ahora, se utilizará la notación Mod_i para denotar el error en distancia del modelo $i \in (1, \ldots, 5)$, $\mathrm{Mod}_{\mathrm{Opt}}$ se referirá al del propagador óptimo y SGP4 al que comete AIDA respecto a SGP4.

A continuación se comparará la capacidad predictiva de los cinco modelos de RN. Para ello, se analiza la precisión de las predicciones desde el inicio de los periodos de entrenamiento, validación y prueba de cada una de las 219 series temporales utilizadas para entrenar los modelos. Los periodos de entrenamiento, validación y prueba se inician en las revoluciones 0, 12 y 14, respectivamente, que corresponden aproximadamente a los días 0, 8 y 10 de la propagación. Inicialmente, se analizan las series que presentaron el mejor y el peor comportamiento para uno de los modelos, seguido de un estudio global del comportamiento de las 219 series para los cinco modelos.

Análisis de las series $\varepsilon^{\theta}_{158}$ y $\varepsilon^{\theta}_{212}$

En este apartado se muestra el comportamiento de las series $\varepsilon^{\theta}_{158}$ y $\varepsilon^{\theta}_{212}$, que fueron las que obtuvieron la mejor y la peor predicción en la fase de entrenamiento del Mod_2, respectivamente.

En la figura **4-1** se compara el comportamiento de la serie temporal $\varepsilon^{\theta}_{158}$, generada a partir del TLE 158, con las predicciones $\widehat{\varepsilon}^{\theta_i}_{158}$ proporcionadas por los cinco modelos de RN. Las dos primeras revoluciones, representadas en línea discontinua al comienzo de los periodos de entrenamiento, validación y prueba, marcan la inicialización del proceso de predicción, mientras que las predicciones, representadas en las gráficas en línea continua, son calculadas utilizando la técnica de la ventana deslizante. En todos los casos, el horizonte de predicción es de 12 días, el cual es equivalente a unas 21 revoluciones del satélite. La evolución de la serie $\varepsilon^{\theta}_{158}$ se muestra en color azul, mientras que las predicciones proporcionadas por los modelos Mod_1 a Mod_5 se representan en los colores marrón, rojo, verde, verde oliva y morado, respectivamente.

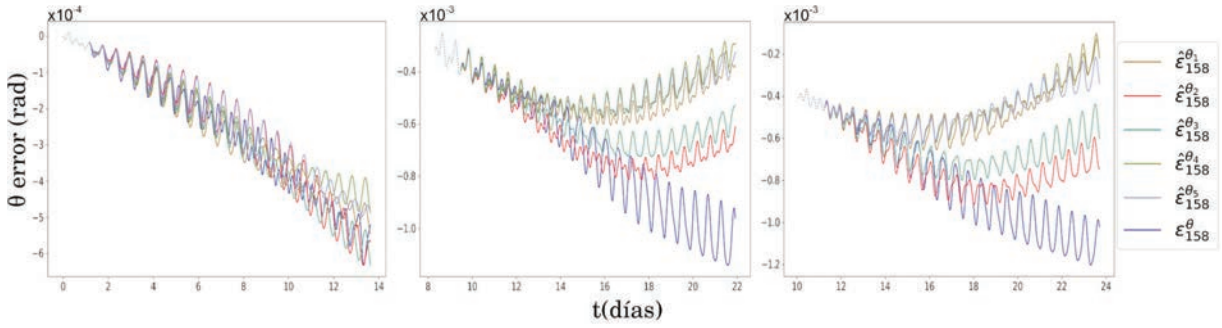


Figura 4-1.: Gráficas de secuencias de las series temporales $\varepsilon^{\theta}_{158}$ y $\widehat{\varepsilon}^{\theta_i}_{158}$, con $i = 1, \ldots, 5$. El comienzo de las predicciones coincide con el inicio de los periodos de entrenamiento (izquierda), validación (centro) y prueba (derecha).

La gráfica de la izquierda en la figura **4-1** evalúa la capacidad de los modelos para reproducir los patrones aprendidos durante la fase de entrenamiento. Como se puede observar, al inicio de la predicción los cinco modelos logran reproducir el comportamiento de la serie temporal. A medida que avanza el tiempo, se detecta un aumento de la amplitud y de la frecuencia de los términos periódicos, aunque los modelos continúan capturando el comportamiento lineal de la tendencia. Aproximadamente a partir del día 9, los modelos Mod_4 y Mod_5 comienzan a mostrar un ligero deterioro en la predicción de la tendencia.

Las tres revoluciones utilizadas durante el proceso de validación de los modelos son parte de los datos representados en la gráfica central de la figura **4-1**. Aunque en las primeras revoluciones las redes neuronales reproducen con bastante fidelidad el comportamiento de la serie $\varepsilon^{\theta}_{158}$, a partir de aproximadamente el cuarto día los modelos Mod_1, Mod_4 y Mod_5 experimentan un deterioro significativo en su comportamiento. Mientras tanto, las predicciones de los modelos Mod_2 y Mod_3 se mantienen cerca de la serie real durante un poco más de tiempo.

En el conjunto de prueba, donde el comportamiento de la serie no ha sido parte del proceso

de entrenamiento, se evalúa la capacidad predictiva del modelo (gráfica de la derecha, figura **4-1**). A corto periodo, los modelos son capaces de predecir el comportamiento de la serie; sin embargo, los modelos Mod_1, Mod_4 y Mod_5 divergen rápidamente. En cambio, los modelos Mod_2 y Mod_3 aproximan bastante bien el comportamiento de la parte estacional y de la tendencia durante aproximadamente 4 días más. Finalmente, solo Mod_2 mantiene una buena predicción durante un periodo un poco más prolongado.

Las tablas **4-5**, **4-6** y **4-7** muestran los errores en distancia entre AIDA y SGP4, Mod_{Opt} y los cinco modelos Mod_i para el TLE 158. En todos los casos, el horizonte de predicción es de 12 días, comenzando al inicio de los periodos de entrenamiento, validación y prueba. Los errores se presentan cada 2, 4, 6, 8, 10 y 12 días. En la primera fila se muestra el error en distancia entre AIDA y SGP4. En la segunda fila aparece el error del mejor modelo predictivo que se podría obtener, Mod_{Opt}. Las cinco filas siguientes representan los errores de los modelos Mod_i.

En la tabla **4-5** se pone de manifiesto que los errores de los cinco modelos son inferiores a los cometidos por SGP4. Después de 12 días de propagación, la magnitud del error varía desde los 0.94 km del Mod_{Opt} hasta los 5.76 km del Mod_4. Es importante destacar que esta tabla cuantifica la información descrita en la figura **4-1**.

Modelo	Error en distancia km – Tiempo en días					
	2	4	6	8	10	12
SGP4	4.26	6.25	7.92	11.16	13.90	17.49
Mod_{Opt}	0.49	0.55	0.55	0.55	0.55	0.94
Mod_1	0.84	2.41	3.52	3.63	3.63	4.04
Mod_2	0.92	1.61	2.83	3.82	3.95	3.95
Mod_3	0.73	2.00	3.04	3.46	3.46	4.36
Mod_4	0.74	1.91	2.58	2.84	2.92	5.76
Mod_5	0.90	1.59	2.71	4.00	4.05	4.05

Tabla 4-5.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de entrenamiento.

En el caso de las predicciones que se inician en el conjunto de validación, los errores de los cinco modelos siguen siendo inferiores a los cometidos por SGP4, como se muestra en la tabla **4-6**. En esta fase, la magnitud del error experimenta un aumento en comparación con la fase de entrenamiento, oscilando entre 1.54 km del Mod_{Opt} y 24.99 km del Mod_4. Los modelos Mod_2 y Mod_3 presentan los errores más bajos, con 13.74 km y 15.89 km, respectivamente, aunque aún distan considerablemente del valor óptimo.

Finalmente, en el conjunto de prueba, el comportamiento de la serie temporal es desconocido para los cinco modelos. Sin embargo, los errores en distancia para todos ellos siguen estando por debajo de los cometidos por SGP4. Aunque, a los 12 días, la variación de los errores con respecto a las predicciones del conjunto de validación solo se incrementan en 3 km, como se puede comprobar en la tabla **4-7**.

En la figura **4-2** se compara el comportamiento de la serie temporal $\varepsilon^{\theta}_{212}$, generada a partir del TLE 212, con las cinco predicciones $\widehat{\varepsilon}^{\theta_i}_{212}$. La serie $\varepsilon^{\theta}_{212}$ es la que presenta una mayor

Modelo	Error en distancia km – Tiempo en días					
	10	12	14	16	18	20
SGP4	14.71	19.14	23.95	29.13	31.85	33.91
Mod_{Opt}	0.55	1.05	1.40	1.54	1.54	1.54
Mod_1	1.46	4.32	7.47	11.50	15.80	24.31
Mod_2	1.47	3.05	3.95	6.40	9.39	13.74
Mod_3	1.13	2.92	4.80	7.88	10.77	15.89
Mod_4	1.48	4.55	7.96	12.77	17.18	24.99
Mod_5	1.41	4.48	8.05	13.28	17.31	22.78

Tabla 4-6.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de validación.

Modelo	Error en distancia km – Tiempo en días					
	12	14	16	18	20	22
SGP4	18.15	23.95	28.36	31.72	33.91	35.94
Mod_{Opt}	0.68	1.39	1.54	1.54	1.54	1.54
Mod_1	1.57	5.20	9.49	15.61	21.62	27.83
Mod_2	0.85	3.21	6.03	9.20	13.64	17.03
Mod_3	1.15	4.31	7.92	11.72	16.31	20.07
Mod_4	1.67	6.00	10.98	16.88	22.81	28.04
Mod_5	1.72	6.79	12.28	17.70	22.60	25.79

Tabla 4-7.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de prueba.

tendencia en la figura **3-8**. Como se aprecia en la figura de la izquierda, los modelos apenas pueden reproducir los patrones observados durante el entrenamiento de las redes neuronales, mientras que, en las etapas de validación (figura central) y prueba (figura derecha), los valores predichos por los modelos se alejan de la serie real desde los primeros instantes.

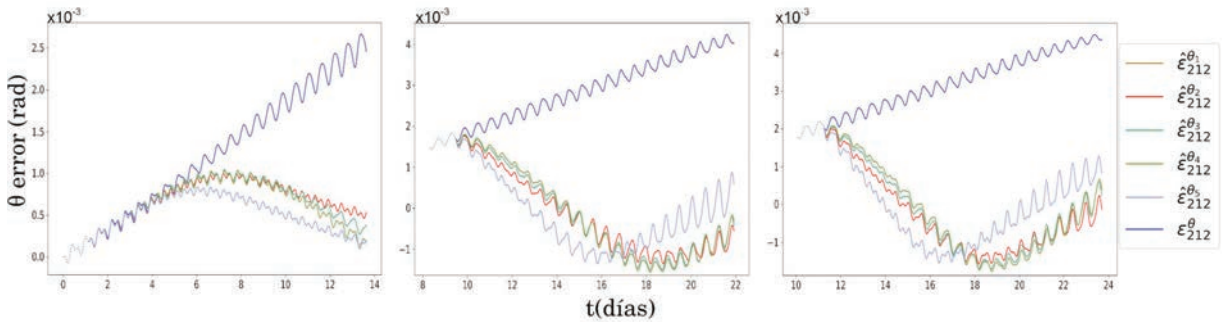


Figura 4-2.: Gráficas de secuencias de las series temporales $\varepsilon_{212}^{\theta}$ y $\widehat{\varepsilon}_{212}^{\theta_i}$, con $i = 1, \ldots, 5$. El comienzo de las predicciones coincide con el inicio de los periodos de entrenamiento (izquierda), validación (centro) y prueba (derecha).

La influencia de las predicciones sobre el error en distancia se muestra en las tablas **4-8**, **4-9** y **4-10**. Al analizar el periodo de entrenamiento (tabla **4-8**) se observa que, aunque el comportamiento se aleja de la serie temporal a partir del cuarto día de propagación, esa pequeña corrección es suficiente para mantener el error por debajo del de SGP4. Sin embargo, en los periodos de validación y prueba (tablas **4-9** y **4-10**, respectivamente), los errores de todos los modelos superan el error cometido por SGP4 a partir del sexto día de propagación.

Modelo	Error en distancia km – Tiempo en días					
	2	4	6	8	10	12
SGP4	16.34	29.03	39.81	52.01	65.38	76.17
Mod_{Opt}	1.03	1.04	1.08	1.34	1.59	1.91
Mod_1	2.73	3.47	8.02	19.03	38.96	61.94
Mod_2	1.18	3.55	11.96	25.31	45.24	60.02
Mod_3	1.91	3.85	11.86	25.35	45.75	64.83
Mod_4	2.22	4.00	12.53	26.61	47.22	68.87
Mod_5	1.57	5.91	17.90	34.53	54.07	69.67

Tabla 4-8.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 212. El comienzo de las predicciones coincide con el inicio del periodo de entrenamiento.

Modelo	Error en distancia km – Tiempo en días					
	10	12	14	16	18	20
SGP4	65.38	79.57	89.81	104.13	114.53	123.33
Mod_{Opt}	1.67	1.98	2.12	2.32	2.37	2.37
Mod_1	18.51	56.66	94.81	139.37	172.29	181.55
Mod_2	31.84	76.03	107.14	133.03	151.64	156.81
Mod_3	27.34	73.90	110.82	145.94	162.12	163.05
Mod_4	25.20	71.24	111.11	148.90	165.27	165.83
Mod_5	41.60	95.36	127.22	137.19	137.19	137.19

Tabla 4-9.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 212. El comienzo de las predicciones coincide con el inicio del periodo de validación.

El resultado poco satisfactorio en el ajuste de los modelos a esta serie puede atribuirse a su marcada tendencia. Como se observa en la figura **3-8**, la serie 212 exhibe una pendiente notablemente más pronunciada en comparación con el resto de series de entrenamiento, lo que la distingue dentro del grupo. Al ser una serie atípica, su comportamiento tiene poca representatividad dentro de la muestra, lo que dificulta al algoritmo de aprendizaje modelarla de manera precisa.

Modelo	Error en distancia km – Tiempo en días					
	12	14	16	18	20	22
SGP4	76.17	89.81	100.69	114.53	123.33	131.52
Mod_{Opt}	1.98	2.12	2.31	2.37	2.37	2.37
Mod_1	28.61	77.28	124.63	174.42	188.62	188.62
Mod_2	47.16	100.20	136.68	157.09	157.77	157.77
Mod_3	41.46	97.88	141.12	167.15	167.30	167.30
Mod_4	37.95	94.28	141.18	171.05	171.86	171.86
Mod_5	58.79	121.96	144.00	144.44	144.44	144.44

Tabla 4-10.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN para un horizonte de propagación de 12 días para el TLE 212. El comienzo de las predicciones coincide con el inicio del periodo de prueba.

Análisis de los modelos de RN. Conjunto de entrenamiento

En este apartado se evalúa la precisión de los modelos de RN con los datos utilizados durante su entrenamiento. Si los modelos no pueden predecir las series temporales con las que fueron entrenados, es probable que no proporcionen predicciones precisas cuando se enfrenten a nuevos comportamientos.

Para ello, se calculará el máximo error en distancia entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de redes neuronales para diferentes horizontes de propagación (2, 4, 6, 8, 10 y 12 días) para cada una de las 219 series. La figura **4-3** muestra los diagramas de caja y bigotes con el análisis de estos máximos. En la tabla **4-11** se presenta el número de valores atípicos. Estos datos no se incluyen en los diagramas para facilitar la visualización de los resultados.

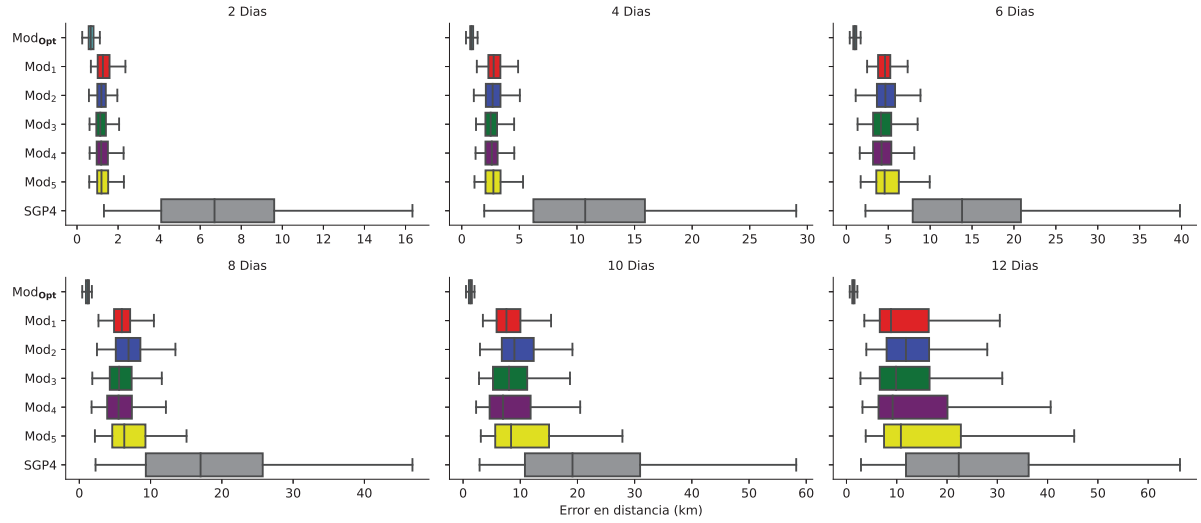


Figura 4-3.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN con los datos de la fase de entrenamiento.

Modelo	Número de valores atípicos – Tiempo en días					
	2	4	6	8	10	12
SGP4	0	0	0	1	1	1
Mod_{Opt}	2	8	0	0	0	0
Mod_1	4	3	5	7	22	20
Mod_2	9	5	6	7	20	22
Mod_3	5	3	1	17	22	22
Mod_4	3	2	3	20	22	16
Mod_5	2	3	6	20	20	13

Tabla 4-11.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-3** según el día de propagación.

Las diferencias entre los bigotes superior e inferior para SGP4 a los 2, 4, 6, 8, 10 y 12 días de propagación son de unos 15.03, 27.08, 37.53, 44.46, 55.49 y 63.44 km, respectivamente. Sin embargo, estas diferencias se reducen a aproximadamente 1.69, 4.21, 8.24, 12.85, 24.76 y 41.46 km, en el caso del máximo de los modelos Mod_i. Es importante destacar que estos valores están significativamente alejados de los 0.91, 1.01, 1.30, 1.36, 1.49 y 1.51 km del Mod_{Opt}. La diferencia con el modelo óptimo indica que hay comportamientos que no han podido ser capturados por ninguno de los modelos de RN. Los máximos errores en distancia cometidos por los modelos Mod_i son de 2.28, 5.32, 9.95, 15.05, 27.86 y 45.29 km para los días 2, 4, 6, 8, 10 y 12, respectivamente. En contraste, los máximos errores alcanzados por SGP4 para esos días son de 16.34, 29.03, 39.81, 46.74, 58.25 y 66.33 km, respectivamente. Por lo tanto, los modelos Mod_i presentan una reducción del error de 14.06, 23.71, 29.86, 31.69, 30.3 y 21.04 km.

El Mod_2 muestra el error más pequeño, con 1.96 km, y la menor dispersión después de 2 días de propagación. A los 4 días, el Mod_3 tiene el error máximo de propagación de 4.54 km. A los 6, 8 y 10 días, el Mod_1 muestra errores de 7.32, 10.47 y 15.39 km, respectivamente. Sin embargo, a los 12 días, el Mod_2 muestra el menor error entre todos los modelos, con 28.0 km. Además, al observar su caja se nota que la mediana del Mod_2 se encuentra aproximadamente en el centro, a diferencia del resto de los modelos Mod_i, que tienden a situarse a la izquierda. Esto indica que Mod_2 presenta una mayor simetría en comparación con los demás modelos. Por otro lado, en el segundo día de propagación, el Mod_1 exhibe el valor máximo más alto y la mayor dispersión, mientras que a partir del cuarto día, el Mod_5 presenta el error máximo más alto y la mayor dispersión.

Se identificaron 42 casos de series que presentaron valores atípicos después de 10 días de propagación (ver tabla **4-11**). De estos, 22 fueron excluidos del análisis debido a que solo son valores atípicos para uno de los modelos. El interés se centró en caracterizar los 20 casos restantes que son comunes a todos los modelos. En la tabla **4-12** se muestran los errores en distancia de estas series, las cuales además fueron clasificadas como de tendencia positiva.

En la figura **4-4** se muestra el comportamiento de las 121 series con tendencia positiva durante los primeros 20 días de propagación. Además, se incluye en color rojo el centroide de estas series, el cual se calcula como la media aritmética de los puntos que ocupan el mismo instante temporal en cada una de las series. Como se puede observar, las series que presentaron valores atípicos en los pronósticos, representadas en diferentes colores, se

$\varepsilon_n^{\theta_i}$	Mod_1	Mod_2	Mod_3	Mod_4	Mod_5
14	20.98	22.98	23.79	26.09	31.25
15	20.25	23.73	24.28	26.07	32.44
23	20.13	23.40	25.10	26.85	32.30
24	20.12	23.38	25.08	26.82	32.28
25	22.33	27.60	28.41	29.58	36.86
32	26.55	31.13	32.62	34.38	40.43
33	30.71	37.64	38.14	38.95	47.09
34	26.94	33.06	33.47	34.48	42.35
40	20.16	24.83	24.85	27.35	33.19
95	20.22	22.71	21.50	23.85	31.04
96	22.35	26.34	24.14	26.07	35.32
97	22.35	26.34	27.30	28.74	35.32
110	27.66	33.41	27.30	28.74	42.40
132	22.42	26.58	21.42	23.74	35.18
180	21.17	24.45	33.94	34.96	33.20
194	24.38	29.20	27.44	28.97	38.30
211	21.36	24.65	25.41	27.00	33.29
212	38.96	45.24	29.93	31.20	54.07
221	23.05	26.58	26.23	27.84	35.27
291	21.66	25.68	45.75	47.22	34.70

Tabla 4-12.: Errores en distancia (km) de las series identificadas como valores atípicos en los modelos Mod_1, Mod_2 y Mod_3 después de 10 días de propagación.

caracterizan por tener una tendencia mayor que la del centroide. De hecho, la pendiente de estas series se encuentra entre 1.56×10^{-6} y 2.55×10^{-6} (ver tabla **4-3**). Este intervalo representa aproximadamente el 16.5 % de las series con pendientes positivas.

Por último, se llevó a cabo un estudio para determinar si los valores atípicos estaban relacionados con los patrones de algunas series en particular. Para ello, las series se dividieron en grupos separados según su similitud utilizando el algoritmo de *deformación dinámica del tiempo rápida* (por su denominación en inglés Fast Dynamic Time Warping, *FastDTW* [171]). En el estudio se analizaron varios casos en los que se solicitó al algoritmo dividir las series en grupos, comenzando desde 2 y aumentando progresivamente hasta llegar a 10 grupos. Sin embargo, no se identificó ninguna relación significativa entre las series que generan valores atípicos en los modelos Mod_i y alguno de los casos estudiados.

El desempeño de los modelos se puede evaluar utilizando el índice de mejora definido en la ecuación (3-4), donde $\mathcal{P} = \text{SGP4}$ y $\mathcal{HP} = \text{Mod}_i$. Este índice, denotado como $\mathcal{I}_{m_{SGP4}}$, varía en el rango de $0 < \mathcal{I}_{m_{SGP4}} < \infty$. Es importante destacar que cuanto más cercano sea el valor del índice a 0, mejor será el desempeño del modelo, lo que indica que tiene una mayor capacidad para reducir el error cometido por SGP4. Los valores cercanos a 1 indican que las predicciones del modelo son similares a las de SGP4. Sin embargo, si el valor supera la unidad, $\mathcal{I}_{m_{SGP4}}$ señala que las predicciones del modelo son menos precisas que las de SGP4.

En la figura **4-5** se muestran los diagramas de caja y bigotes que analizan $\mathcal{I}_{m_{SGP4}}$ para

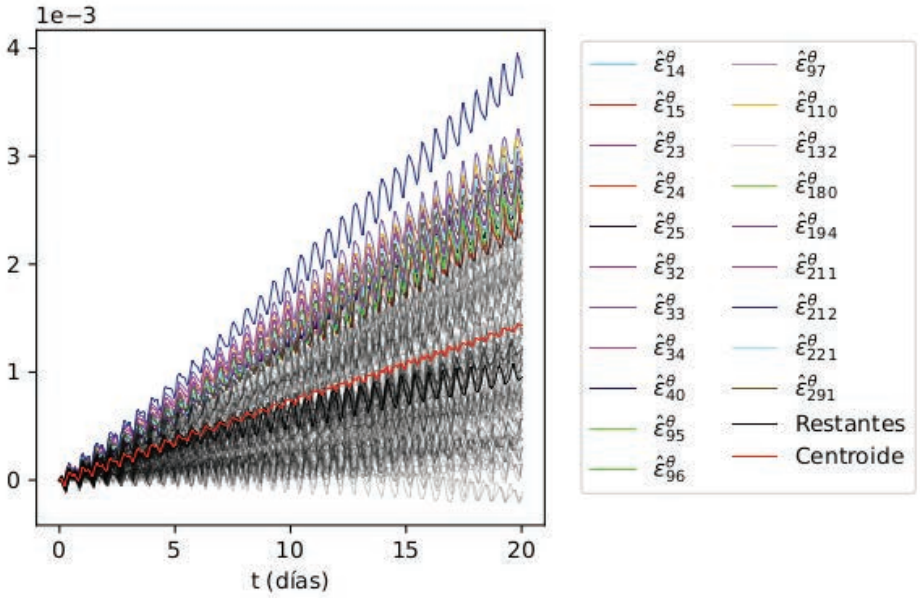


Figura 4-4.: Gráfico de secuencias de las 121 series temporales con tendencia positiva del conjunto de entrenamiento. En rojo se muestra el centroide de todas las series; los 20 valores atípicos, presentes en los cinco modelos, se destacan con colores diferentes, mientras que las 99 series restantes se representan en gris.

los cinco modelos de RN durante el periodo de entrenamiento. La Tabla **4-13** proporciona el número de casos en los que las predicciones de los modelos empeoraron en comparación con SGP4, es decir, cuando $\mathcal{I}_{m_{SGP4}} > 1$.

El Mod_2, que exhibió los errores en distancia más pequeños después de 2 días de propagación, muestra una diferencia entre los extremos de los bigotes de aproximadamente 0.54 unidades, siendo estos valores los más altos después del Mod_5. Sin embargo, los modelos Mod_1, Mod_3, y Mod_4 logran reducir esta diferencia en mayor proporción, a pesar de presentar errores en distancia mayores que los del Mod_2. Durante este periodo, todos los resultados de SGP4 fueron mejorados.

El Mod_3 mostró la menor diferencia entre los extremos de los bigotes de todos los modelos, aproximadamente 0.83 unidades hasta los 4 días. En este día, el modelo identificó 5 valores atípicos, siendo el más alto 1.34 unidades correspondiente a la serie ϵ^{θ}_{62} clasificada como sin pendiente, con un valor de 7.85×10^{-08} (ver tabla **4-3**). En esta predicción, el error se incrementó de 12.10 a 16.21 km, y el modelo no logró reproducir adecuadamente la amplitud de las oscilaciones de la serie. En 9 casos, los resultados del modelo presentaron errores mas grandes que SGP4.

A los 6, 8 y 10 días el Mod_1 muestra una diferencia entre los extremos de los bigotes de 1.15 unidades. Durante este período, en Mod_1 se registraron 7, 8 y 10 valores atípicos respectivamente. El valor más alto, 2.31 unidades, se observó el día 10, que corresponde a la serie ϵ^{θ}_{19} clasificada también como sin pendiente, con un valor de 5.43×10^{-08}. El error de la

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	2	4	6	8	10	12
Mod_1	0	5	12	22	26	26
Mod_2	0	9	21	33	43	44
Mod_3	0	5	14	19	30	40
Mod_4	0	5	13	15	19	25
Mod_5	0	7	21	31	34	31

Tabla 4-13.: Número de casos en los que las predicciones de los modelos Mod_i empeoraron en comparación con SGP4 durante el periodo de entrenamiento.

predicción a los 12 días se incrementó entonces desde los 7.68 hasta los 17.74 km. En los días 10 y 12 se obtuvieron el mismo número de valores atípicos, 26, en los cuales las predicciones con este modelo empeoraron los resultados de SGP4.

Por último, a los 12 días de propagación, el Mod_2 empeoró las predicciones de SGP4 en 44 casos. Además, para este día, el modelo registra la diferencia más alta entre los bigotes en el análisis del $\mathcal{I}_{m_{SGP4}}$, con 1.43 unidades.

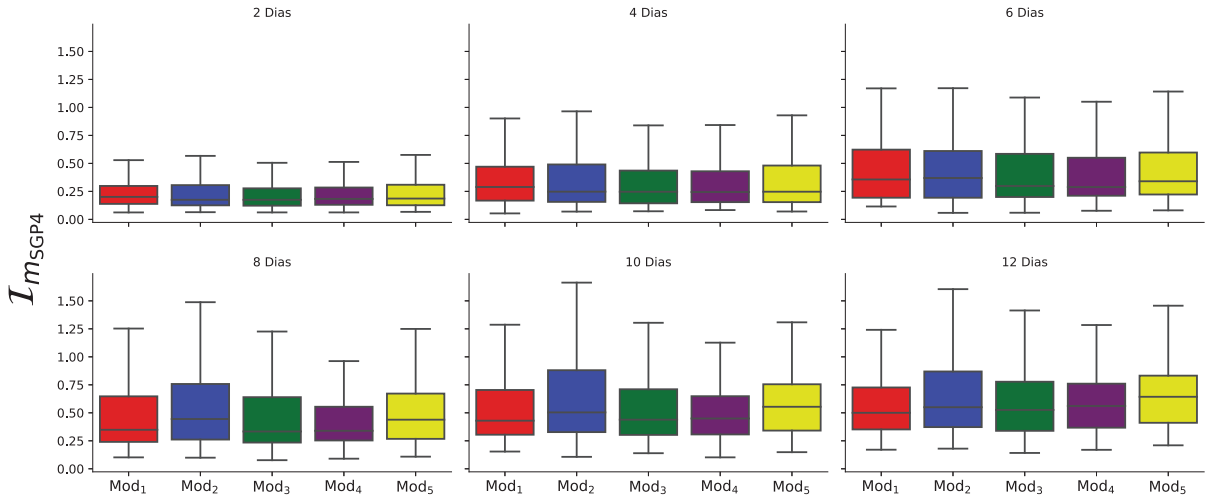


Figura 4-5.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto a SGP4 para los modelos Mod_i, durante los 12 días de propagación con los datos de entrenamiento.

Se identificó un conjunto de 26 series en las cuales los resultados de todos los modelos Mod_i empeoraron las predicciones de SGP4 en al menos uno de los 12 días de propagación. Al clasificar las series según su tendencia se encontró que 11 tenían tendencia positiva, 4 negativa y 11 fueron etiquetadas como sin tendencia. Es importante recordar que de las 219 series de entrenamiento, solo 18 fueron clasificadas como series sin tendencia. Por lo tanto, es de esperar que los modelos no cuenten con suficiente información para aprender los patrones de este tipo de series. En la figura **4-6** se muestran las 219 series temporales utilizadas en la fase de entrenamiento. Se destacan en color rojo las 26 series en las que los modelos empeoran los resultados de SGP4 ($\mathcal{I}_{m_{SGP4}} > 1$).

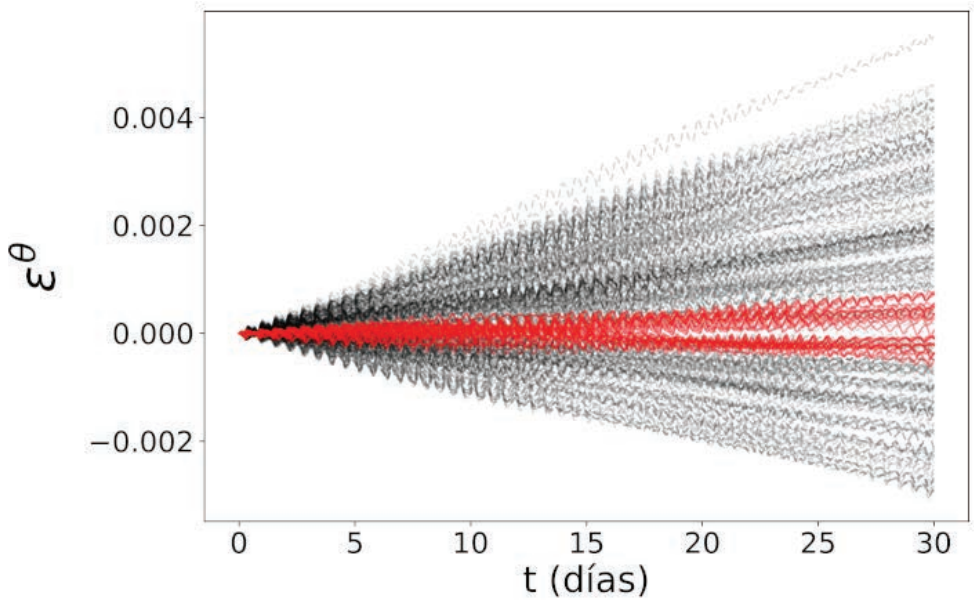


Figura 4-6.: Gráfico de secuencias de las 219 series temporales del conjunto de entrenamiento. Las 26 series en las que los modelos empeoraron los resultados de SGP4 se destacan en color rojo, mientras que las series restantes se representan en gris.

La serie ε^θ_{282} presenta una tendencia de 4.23×10^{-07} unidades, que es la tendencia positiva más grande entre las 26 series (ver tabla **4-3**). El límite superior empleado para clasificar una serie como sin tendencia es de 8.78×10^{-08} unidades, lo que implica una diferencia de 3.35×10^{-07} unidades. Por otro lado, la diferencia entre la serie con la tendencia negativa más pronunciada de las 26 series (ε^θ_{100}) y el límite inferior para clasificar una serie como sin tendencia es de 3.55×10^{-07} unidades. Esto indica que las series en las que los modelos Mod_i empeoran los resultados de SGP4 tienden a ser clasificadas como sin tendencia, o bien, tienen una tendencia tan poco pronunciada que estuvieron cercanas a ser clasificadas como tales. De las 18 series sin tendencia, ε^θ_{67}, ε^θ_{68}, ε^θ_{105} y ε^θ_{164} presentaron índices de mejora menores a la unidad en al menos tres de los modelos híbridos. Por ejemplo, para el modelo Mod_2, los índices de mejora de estas series fueron de 0.87, 0.87, 0.81 y 0.66 unidades para el día 12. Esto redujo el error de predicción de SGP4 para cada una de estas series de 4.77, 4.77, 10.25 y 8.72 km a 4.14, 4.14, 8.30 y 5.75 km, respectivamente.

Es importante destacar que el índice $\mathcal{I}_{m_{SGP4}}$ es un valor relativo, lo que significa que un índice alto no necesariamente indica un error de predicción del modelo significativamente alto. Por ejemplo, el índice de mejora para la serie ε^θ_{19} con el Mod_1 es de 2.34 unidades a los 10 días de propagación. Sin embargo, para esta serie el error en distancia de Mod_1 es de 16.30 km, mientras que el de SGP4 es de 7.68 km lo que implica un incremento del error de solo 8.62 km.

Análisis de los modelos de RN. Conjunto de validación

La figura **4-7** presenta los diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los modelos de RN para cada uno de los 219 TLE a partir del décimo día de propagación. El número de valores atípicos se muestra en al tabla **4-14**. Estos valores no se incluyen en los diagramas, al igual que en el análisis realizado sobre el conjunto de entrenamiento.

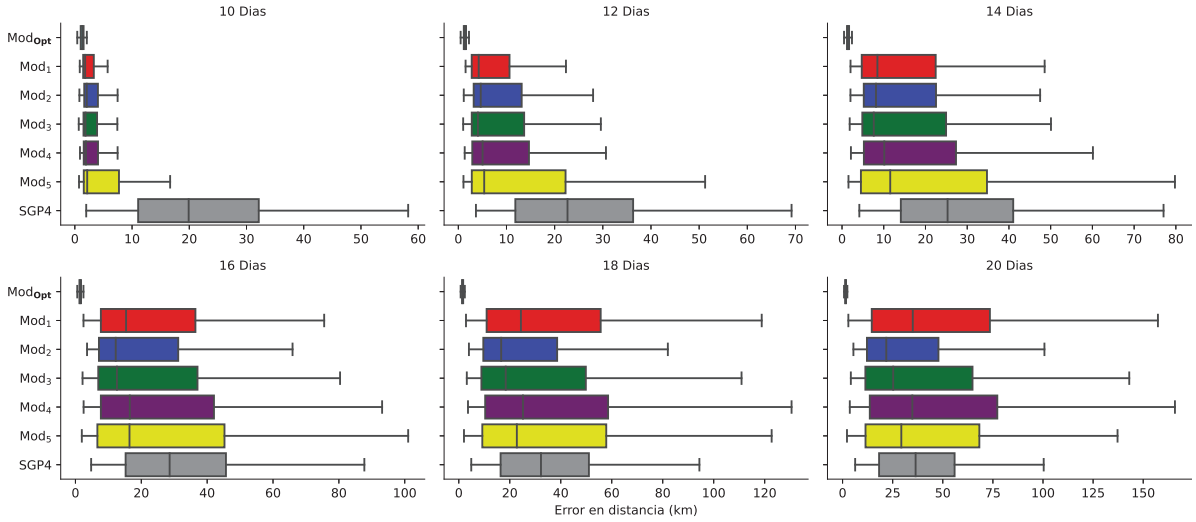


Figura 4-7.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN. Las propagaciones comienzan en la fase de validación.

Modelo	Número de valores atípicos – Tiempo en días					
	10	12	14	16	18	20
SGP4	1	1	1	1	1	1
Mod_{Opt}	0	1	0	0	5	4
Mod_1	25	21	15	15	6	1
Mod_2	35	24	20	19	17	14
Mod_3	28	22	20	15	12	1
Mod_4	26	20	14	9	4	0
Mod_5	23	13	12	9	1	0

Tabla 4-14.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-7** según el día de propagación.

La variación entre los extremos superior e inferior de los bigotes para SGP4 es de 56.26, 65.53, 72.90, 82.96, 89.49 y 94.07 km para los días 10, 12, 14, 16, 18 y 20 de propagación, respectivamente. En el caso del Mod_5 que presentó los perores resultados entre el día 10 y 16 y Mod_4 que presentó los peores resultados durante los días restantes, esta diferencia es de 16.35, 50.17, 78.63, 102.02, 126.93 y 162.23 km respectivamente. Es decir, se observa una

reducción del error de SGP4 de aproximadamente 16.48 y 50.11 km para los días 10 y 12, pero un incremento de 77.68, 101.01, 126.62 y 161.26 km para los días restantes. En el día 10, la diferencia entre los extremos de los bigotes es inferior a 10 km, y el error máximo no supera el primer cuartil de los errores de SGP4 (ubicado alrededor de los 11.14 km) para los cuatro primeros modelos. En el caso del Mod_5, es inferior a la mediana (unos 19.95 km). No obstante, estos valores difieren significativamente de los 1.67 km del error máximo mostrado para este día por parte del Mod_{Opt}.

A medida que avanza el tiempo de propagación, el error máximo registrado por los modelos Mod_i es de 17.08, 51.24, 80.18, 103.96, 130.54 y 165.83 km para los días 10, 12, 14, 16, 18 y 20, respectivamente. Estos errores se acercan a los máximos obtenidos por SGP4 durante los días 2 y 4, de 58.25 y 69.19 km, respectivamente, e incluso los superan durante los días restantes, donde SGP4 registró errores máximos de 77.05, 87.73, 94.38 y 100.35 km. Los errores más pequeños, de 5.73 y 22.35 km, fueron alcanzados por Mod_1 en los días 12 y 14, respectivamente. En los días siguientes, los errores más bajos fueron registrados por Mod_2, con valores de 47.47, 68.23, 82.01 y 100.71 km, respectivamente.

Durante los dos primeros días de propagación se alcanzó una concentración significativa de valores atípicos en todos los modelos. Por ejemplo, en el día 10, Mod_2 registró 35 valores atípicos, lo que representa aproximadamente el 16 % de sus predicciones. Sin embargo, a medida que transcurre el tiempo de propagación, se observa una disminución en la cantidad de valores atípicos, en los modelos restantes.

La figura **4-8** ilustra el análisis de $\mathcal{I}_{m_{SGP4}}$ durante el periodo de validación, mientras que la tabla **4-15** muestra el número de casos en los cuales las predicciones de los modelos resultaron ser peores que las de SGP4.

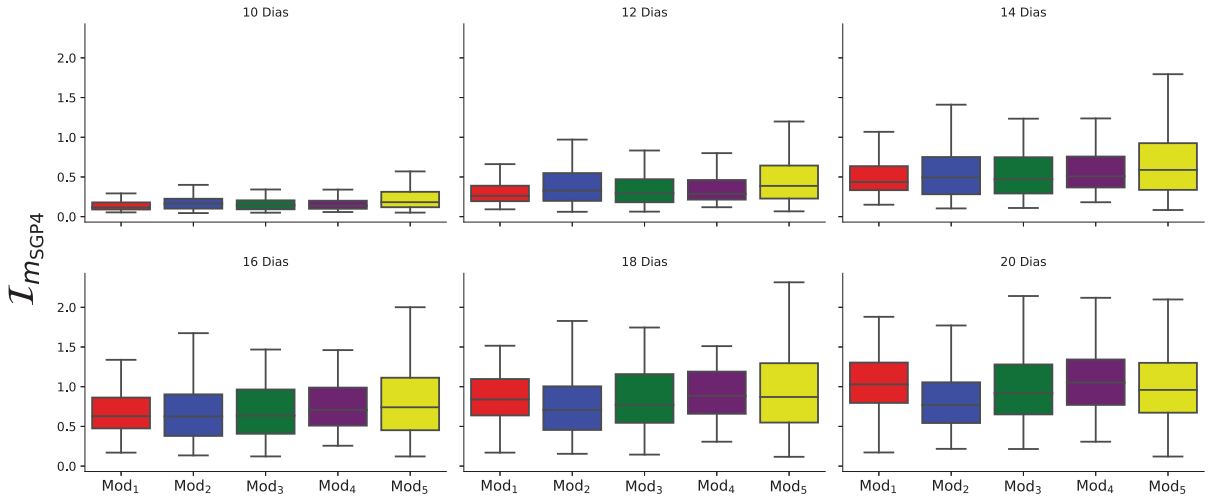


Figura 4-8.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto a SGP4 para los modelos Mod_i, para un horizonte de propagación de 10 días con los datos de validación.

Destaca el Mod_5, que mostró el peor rendimiento en el día 10 y la mayor diferencia entre los extremos de los bigotes, aproximadamente 0.55 unidades. Por otro lado, la diferencia

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	10	12	14	16	18	20
Mod_1	0	0	5	26	74	114
Mod_2	1	3	16	41	56	74
Mod_3	0	1	11	48	73	94
Mod_4	0	0	12	53	86	126
Mod_5	0	7	48	68	91	102

Tabla 4-15.: Casos en los que las predicciones de los modelos Mod_i empeoraron en comparación con SGP4 durante el periodo de validación.

entre los bigotes para los cuatro modelos restantes es menor a 0.36 unidades durante estos dos primeros días, mejorando todos los resultados de SGP4. No obstante, se observó una excepción en el caso del Mod_2, que mostró 7 valores atípicos, uno de ellos superando la unidad. Hasta el día 12, los modelos Mod_1 y Mod_4 mejoraron los resultados de SGP4, con diferencias entre los bigotes de 0.59 y 0.71 unidades, respectivamente. Cada uno de estos modelos presentó 5 y 4 valores atípicos, todos con un índice inferior a 1. En el día 20 de propagación, el Mod_2 mostró la menor diferencia entre los bigotes, 1.6 unidades, entre todos los modelos de RN. A pesar de que 74 de las predicciones realizadas por este modelo resultaron ser peores que las de SGP4, es importante destacar que sigue siendo el modelo con el mejor desempeño. En este día, se registraron 17 valores atípicos, siendo el más alto de 3.82 unidades, correspondiente a la serie sin tendencia $\varepsilon^{\theta}_{226}$, donde el error de predicción se incrementó de 6.40 a 24.12 km.

Análisis de los modelos de RN. Conjunto de prueba

La figura **4-9** muestra los diagramas de caja y bigotes que representan los máximos de los errores en distancia (km) entre AIDA y SGP4, así como entre AIDA y los modelos Mod_{Opt} y Mod_i para cada uno de los 219 TLE a partir del duodécimo día de propagación. El número de valores atípicos se presenta en la tabla **4-16**. Estos valores no se incluyen en los diagramas, al igual que en el análisis realizado en los conjuntos de entrenamiento y validación.

Modelo	Número de valores atípicos – Tiempo en días					
	12	14	16	18	20	22
SGP4	1	1	1	1	1	1
Mod_{Opt}	0	1	0	3	1	2
Mod_1	25	22	15	10	3	14
Mod_2	24	23	19	15	15	0
Mod_3	24	21	15	9	3	0
Mod_4	24	20	14	7	1	0
Mod_5	18	14	10	1	0	0

Tabla 4-16.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-9** según el día de propagación.

La diferencia entre los bigotes superior e inferior para SGP4 es de 63.42, 73.53, 81.31, 89.67,

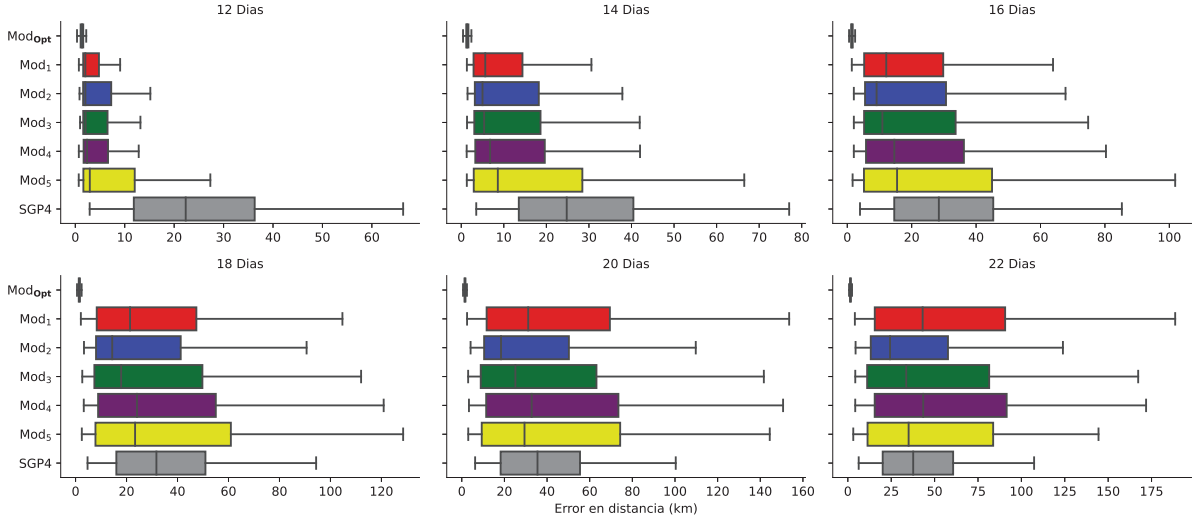


Figura 4-9.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y los cinco modelos de RN. Las propagaciones comienzan en la fase de prueba.

94.09 y 101,07 km para los días 12, 14, 16, 18, 20 y 22, respectivamente. En cambio, en el caso del peor de los modelos entre los días 12 a 18, Mod$_5$, estas diferencias son de aproximadamente 26.62, 65.19, 104.40 y 126.07 km, y en Mod$_1$ que presentó los peores resultados los días restantes, esta diferencia es de 151.07 y 184.57 km. Es decir, la dispersión de SGP4 se ha reducido en al menos 36.8 y 8.33 km para los días 12 y 14, mientras que para el resto de los días, se ha incrementado en 23.09, 36.39, 56.98 y 83.50 km. Para los cuatro primeros modelos, la diferencia para el día 12 es menor a 16.00 km, lo cual, aunque es un resultado aceptable, está considerablemente distante de los 1.90 km del error máximo del Mod$_{\text{Opt}}$. A medida que avanza la propagación, los errores máximos registrados por los modelos Mod$_i$ son de unos 27.32, 66.50, 106.06 y 128.52 km entre los días 12 y 18, respectivamente, para el Mod$_5$, y de 153.55 y 188.62 km para los días restantes, en el caso del Mod$_1$. Estos errores se acercan a los 66.33, 77.05 y 85.26 km observados por SGP4 del día 12 al 14, respectivamente, e incluso los superan en los días siguientes, donde los errores de SGP4 alcanzan los 94.38, 100.35 y 107.33 km. Por otra parte, el Mod$_1$ presenta los errores más pequeños, con valores de 9.92, 31.86 y 63.89 km durante los días 12, 14 y 16, respectivamente, mientras que el Mod$_2$ registra errores de 90.64, 109.74 y 126.84 km en los días siguientes.

En este caso, el número de valores atípicos durante los primeros 2 días de propagación, es decir, al día 12, es similar al detectado en el conjunto de validación en el día 10. El Mod$_1$ tiene el mayor número de valores atípicos, con un total de 25, mientras que el Mod$_5$ muestra el menor número, con solo 18.

El análisis de $\mathcal{I}_{m_{SGP4}}$ para los datos del conjunto de prueba y una propagación de 12 días se muestra en la figura **4-10** mediante diagramas de caja y bigotes. La mayor diferencia entre el bigote superior e inferior la presenta Mod$_2$, con aproximadamente 2.3 unidades. Este modelo registra 4 valores atípicos, siendo el más elevado de 3.8 unidades, el cual corresponde a la

serie sin tendencia $\varepsilon^{\theta}_{226}$. En esta serie el error se incrementó de 6.40 a 24.46 km. A pesar de esto, Mod_2 posee el menor valor del cuartil Q_3 entre todos los modelos, con tan solo 1.3 unidades.

Al igual que en el conjunto de validación, el índice de mejora mínimo se encuentra en Mod_5, con un valor de aproximadamente 0.16 unidades, mientras que en los demás modelos, este mínimo se sitúa en promedio alrededor de las 0.28 unidades. Aparte del Mod_2, solo se registró un valor atípico en el Mod_5 de 3.29 unidades, también relacionado con la serie $\varepsilon^{\theta}_{226}$, donde el aumento del error es de 14.65 km. El número de casos en los cuales las predicciones de los modelos Mod_i empeoraron respecto a las predicciones de SGP4 fueron de 139, 80, 105, 133 y 118, respectivamente. Es evidente que con tan solo 80 casos, Mod_2 presenta el mejor comportamiento de todos.

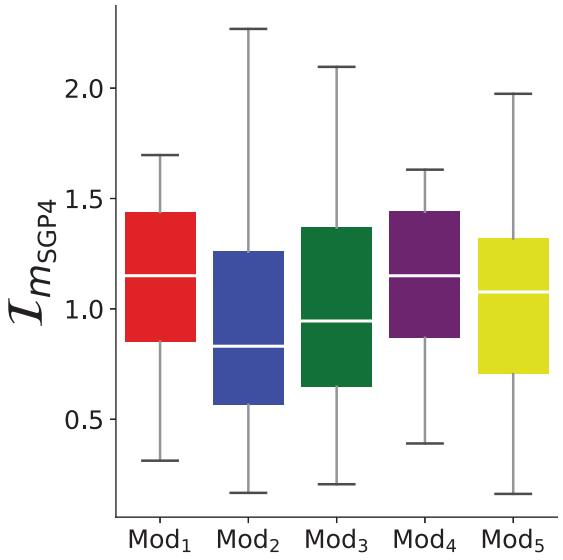


Figura 4-10.: Diagramas de caja y bigotes que muestran el análisis del índice de mejora con respecto a SGP4 para los modelos Mod_i, con un horizonte de propagación de 12 días y utilizando datos de prueba.

Mejor modelo

El proceso de selección del modelo más adecuado está determinado por el período de propagación deseado. En situaciones que requieran una precisión particularmente alta a corto plazo, Mod_1 se posiciona como una opción destacada. Este modelo muestra la menor diferencia entre los bigotes superior e inferior, así como el valor más bajo de Q_3 (10.02 km) entre todos los modelos durante los primeros 10 días de propagación con los datos de entrenamiento. En la fase de validación, Mod_1 también mantiene la menor diferencia entre los bigotes y el Q_3 más bajo (10.63 km) durante los primeros 4 días, donde ninguno de sus predicciones produjo una disminución de la precisión en comparación con SGP4. Al realizar

predicciones en los datos de prueba, el modelo continúa mostrando los mejores resultados durante los primeros 4 días de propagación, donde el valor de Q_3 alcanza los 14.38 km.

Cuando se busca mejorar la precisión a largo plazo, es crucial que las predicciones sean estables durante un período prolongado. En este escenario, Mod_2 destaca por su consistente rendimiento. En los diagramas de dispersión con los datos de entrenamiento, se observa que a los 12 días de propagación, el valor de Q_3 (16.43 km) es el más bajo en comparación con los otros modelos. Además, al realizar la propagación desde los días 10 y 12, este modelo muestra el valor más bajo de Q_3 a partir de los días 16 y 18 con valores de 31.21 y 30.66 km, respectivamente. El índice $\mathcal{I}_{m_{SGP4}}$ indica que este modelo mejora la precisión de las predicciones proporcionadas por SGP4 en al menos un 75 % de los casos al día 12 de propagación en el conjunto de entrenamiento. En los conjuntos de validación y prueba, el índice muestra que el porcentaje de mejora oscila entre el 50 % y el 75 %. A pesar de ello, este porcentaje de mejora es el más alto entre todos los modelos en estos períodos.

El índice de mejora respecto al óptimo, definido por la ecuación (3-5), permite comparar el desempeño de los modelos de RN en relación con el rendimiento del modelo óptimo. En la figura **4-11** se presentan los diagramas de caja y bigotes que analizan $\mathcal{I}_{m_{\mathrm{Mod_{Opt}}}}$ para los cinco modelos durante un horizonte de propagación de 12 días, utilizando los datos de los conjuntos de entrenamiento, validación y prueba. Los valores de este índice están en el rango de 0 a 1, donde un valor cercano a 1 indica que las predicciones de los modelos se acercan más a las predicciones óptimas. Observando la figura, se puede ver que con los datos de entrenamiento, Mod_2 muestra el rendimiento más bajo entre los cinco modelos, ya que tanto la media como el Q_3 son más pequeños. Sin embargo, en los conjuntos de validación y prueba, presenta un rendimiento similar al del resto de los modelos. En todos los casos la posibilidad de mejora es muy alta.

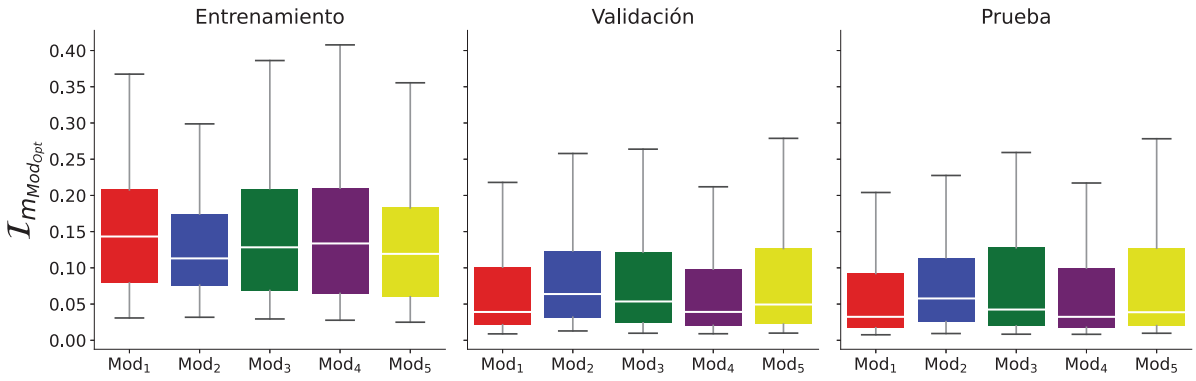


Figura 4-11.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto al óptimo $\mathcal{I}_{m_{\mathrm{Mod_{Opt}}}}$ para los modelos Mod_i, con los datos de entrenamiento, validación y prueba, y un horizonte de propagación de 12 días.

4.2.4. Modelo aleatorio

Una de las conclusiones extraídas del análisis de los modelos previos es que el orden en el que se preparan los datos, aunque las series no están ordenadas por tendencia, introduce un sesgo en el proceso de aprendizaje. Esto puede afectar a la capacidad de generalización y potencialmente obstaculizar el proceso de regularización que se está utilizando (Early stopping). En consecuencia, parece que los modelos están aprendiendo patrones específicos relacionados con el orden de los datos.

El uso de mezcla aleatoria, también conocida como *shuffle* en inglés [172, 173], en el preprocesamiento del conjunto de datos de entrenamiento, es una práctica común en el tratamiento de imágenes. Esta técnica ayuda a evitar sesgos de aprendizaje al exponer la red a muestras de diferentes series y distribuciones en cada época de entrenamiento. Además, contribuye a mejorar la capacidad de generalización del modelo con respecto a nuevos datos.

A continuación, se volverán a entrenar cinco modelos de RN utilizando la mezcla aleatoria. En este caso, una vez que se han construido los 3,811,401 vectores del conjunto de entrenamiento y los 820,999 vectores del conjunto de validación, se procede a mezclarlos de forma aleatoria utilizando la función Shuffle implementada en *Tensorflow*. Esta función utiliza el algoritmo Philox [174] para generar números aleatorios. De esta manera, se evita que los vectores correspondientes a una misma serie se introduzcan consecutivamente en la RN, lo que permite que el ajuste de los pesos de la red neuronal se realice con muestras provenientes de cualquier serie y en cualquier instante temporal.

En la tabla **4-17** se muestra el tiempo de cómputo empleado para entrenar los cinco modelos. Cada uno de ellos fue ejecutado en Beronia utilizando 28 CPU. Los modelos Mod_3, Mod_4 y Mod_5 presentan tiempos similares, alrededor de 32 días. Sin embargo, el entrenamiento de estos modelos se detuvo al alcanzar el límite del número de épocas permitido (6000). Esta interrupción indica que el número de épocas de entrenamiento no fue suficiente para que el proceso de optimización finalizara con éxito, por lo que estos modelos se descartan en los análisis posteriores.

Modelo	Tiempo (días)
1	26.32
2	22.58
3	31.76
4	32.05
5	32.02

Tabla 4-17.: Tiempo de cómputo de los cinco modelos de RN.

A continuación, se compara la capacidad predictiva de los dos primeros modelos. Al igual que en el apartado 4.2.3, se analiza la precisión de las predicciones desde el inicio de los periodos de entrenamiento, validación y prueba de cada una de las 219 series temporales utilizadas para entrenar los modelos. Los periodos de entrenamiento, validación y prueba comienzan en las revoluciones 0, 12 y 14, respectivamente, que corresponden a los días 0, 8 y 10 de la propagación. En primer lugar, se presenta un análisis detallado de las series que

exhibieron el mejor y el peor comportamiento para el Mod_1, seguido de un estudio global del comportamiento de las 219 series para los dos modelos.

Análisis de las series $\varepsilon^{\theta}_{279}$ y $\varepsilon^{\theta}_{80}$

En este apartado se analiza el comportamiento de las series $\varepsilon^{\theta}_{279}$ y $\varepsilon^{\theta}_{80}$, las cuales registraron la mejor y la peor predicción durante la fase de entrenamiento del Mod_1.

La serie temporal $\varepsilon^{\theta}_{279}$ se compara con las predicciones proporcionadas por los dos modelos de RN para un horizonte de predicción de 12 días en la figura **4-12**. Al igual que en el caso secuencial, la línea discontinua representa la inicialización del proceso. Esta serie se representa en color azul, mientras que las predicciones de los modelos $\widehat{\varepsilon}^{\theta_1}_{279}$ y $\widehat{\varepsilon}^{\theta_2}_{279}$ se muestran en color rojo y verde, respectivamente. En la gráfica de la izquierda de la figura **4-12**, se observa que ambos modelos reproducen con bastante fidelidad la tendencia y la frecuencia de los términos periódicos de la serie temporal durante los primeros 12 días. Sin embargo, ninguno de ellos logra predecir la amplitud de dichos términos, la cual aumenta a medida que avanza el tiempo.

Sobre el conjunto de validación (gráfica central de la figura **4-12**), las RN reproducen inicialmente tanto la tendencia como las oscilaciones de la serie. No obstante, a partir del cuarto día de propagación, la serie $\varepsilon^{\theta}_{279}$ experimenta un cambio en la tendencia y la forma de las oscilaciones. Este cambio logra ser capturado por el modelo Mod_1 durante aproximadamente 2 días, momento en el cual la precisión de la predicción comienza a deteriorarse. Por otro lado, Mod_2 no consigue reproducir estos cambios, por lo que sus predicciones se deterioran antes.

En el conjunto de prueba (gráfica de la derecha, figura **4-12**) los modelos son capaces de predecir el comportamiento de la serie durante los primeros 6 días, momento en el cual se aprecia un cambio en la tendencia, la amplitud y la frecuencia de los términos periódicos. Estos cambios no son predichos por los modelos, lo que provoca que el comportamiento de las estimaciones diverja rápidamente. Sin embargo, Mod_1 logra capturar la tendencia real de la serie por un período de tiempo mayor que Mod_2.

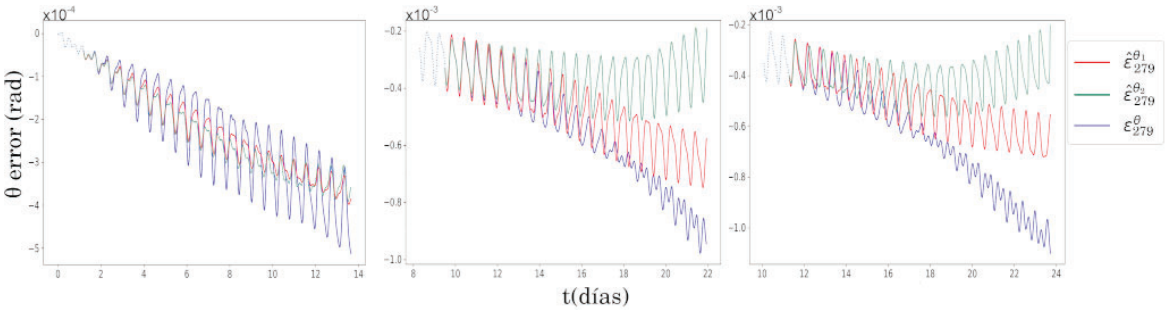


Figura 4-12.: Gráficas de secuencias de las series temporales $\varepsilon^{\theta}_{279}$, $\widehat{\varepsilon}^{\theta_1}_{279}$ y $\widehat{\varepsilon}^{\theta_2}_{279}$. El comienzo de las predicciones coincide con el inicio de los periodos de entrenamiento (izquierda), validación (centro) y prueba (derecha).

Las tablas **4-18**, **4-19** y **4-20** muestran los errores en distancia entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 279. Al igual que en el caso de los modelos

secuenciales, el horizonte de predicción es de 12 días comenzando al inicio de los periodos de entrenamiento validación y prueba. Los errores que aparecen en las tablas corresponden a los días 2, 4, 6, 8, 10 y 12 de propagación.

Los errores en distancia entre AIDA y SGP4 en el conjunto de entrenamiento (ver tabla **4-18**) varían desde 4.82 km a los 2 días de propagación hasta 13.89 km a los 12 días. En el mismo intervalo, los errores del Mod_{Opt} van desde 0.50 km hasta 1.72 km. Sin embargo, al cabo de 12 días, los errores en distancia del Mod_1 y Mod_2 son ligeramente superiores a los del Mod_{Opt}, registrando 2.95 km y 3.39 km, respectivamente. Cabe destacar que estos valores son significativamente menores que los 13.89 km observados en SGP4.

Modelo	Error en distancia km – Tiempo en días					
	2	4	6	8	10	12
SGP4	4.82	7.05	9.37	10.94	12.04	13.89
Mod_{Opt}	0.50	0.82	1.20	1.40	1.58	1.72
Mod_1	0.66	1.30	2.10	2.37	2.59	2.95
Mod_2	0.80	1.89	2.79	2.84	3.05	3.39

Tabla 4-18.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de entrenamiento.

En el conjunto de validación (consultar tabla **4-19**), los errores de ambos modelos continúan siendo inferiores a los errores de SGP4. Destaca que los errores del Mod_2 experimentan un notable aumento a partir del día 12. En el día 20, el error del Mod_1 es de 9.23 km, mientras que el error del Mod_2 es de 22.47 km. Estos valores difieren considerablemente de los 1.76 km del Mod_{Opt} y representan un incremento significativo en comparación con la fase de entrenamiento.

Modelo	Error en distancia km – Tiempo en días					
	10	12	14	16	18	20
SGP4	12.27	13.89	16.14	18.37	22.32	27.00
Mod_{Opt}	1.62	1.76	1.76	1.76	1.76	1.76
Mod_1	1.74	1.86	2.48	4.62	6.43	9.23
Mod_2	1.55	1.99	5.11	10.39	14.38	22.47

Tabla 4-19.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de validación.

Por último, en el conjunto de prueba (ver tabla **4-20**), los errores en distancia de los modelos continúan siendo inferiores a los errores de SGP4, aunque son considerablemente superiores al modelo óptimo. A pesar de ello, muestran un incremento con respecto al conjunto de validación de aproximadamente 6 km a los 12 días de propagación (día 22 en la gráfica).

En la figura **4-13** se compara el comportamiento de la serie temporal $\varepsilon_{80}^{\theta_i}$, generada a partir del TLE 80, con las predicciones generadas por los modelos Mod_1 y Mod_2. Como se

Modelo	Error en distancia km – Tiempo en días					
	12	14	16	18	20	22
SGP4	13.89	15.51	18.37	22.32	27.00	30.80
Mod_{Opt}	1.76	1.76	1.76	1.76	1.76	1.76
Mod_1	1.83	2.60	4.25	6.55	8.84	14.68
Mod_2	1.74	4.07	7.67	12.75	18.23	28.98

Tabla 4-20.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 158. El comienzo de las predicciones coincide con el inicio del periodo de prueba.

observa en la figura de la izquierda, ninguno de los modelos logra reproducir con precisión la tendencia de la serie. Además, se aprecia una variación en la amplitud y frecuencia de los términos periódicos que no es capturada por ninguno de los modelos. Durante las fases de validación (figura central) y prueba (figura derecha), Mod_1 predice la tendencia de la serie, así como los términos periódicos, mostrando un comportamiento estable. Por otro lado, Mod_2 exhibe un comportamiento inestable, lo que provoca que sus predicciones se desvíen de los datos reales a partir del sexto día de propagación aproximadamente, tanto en el conjunto de validación como en el de prueba.

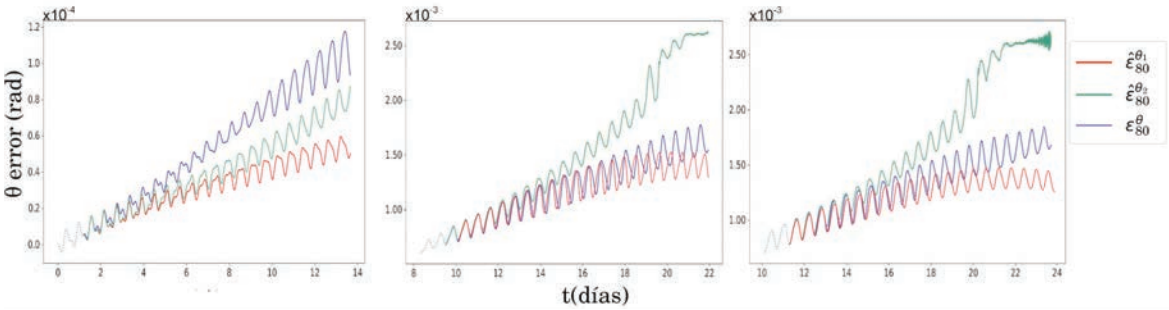


Figura 4-13.: Gráficas de secuencias de las series temporales $\varepsilon_{80}^{\theta}$, $\widehat{\varepsilon}_{80}^{\theta_1}$ y $\widehat{\varepsilon}_{80}^{\theta_2}$. El comienzo de las predicciones coincide con el inicio de los periodos de entrenamiento (izquierda), validación (centro) y prueba (derecha).

Las tablas **4-21**, **4-22**, y **4-23** muestran los errores en distancia entre AIDA y SGP4, junto con los generados por los tres modelos del tipo de Encke. En el periodo de entrenamiento (tabla **4-21**), la corrección proporcionada por los modelos es suficiente para mantener el error por debajo de los errores mostrados por SGP4 durante todo el tiempo de propagación. No obstante, aún se observa que los errores están significativamente por encima de los valores obtenidos por Mod_{Opt}.

En los conjuntos de validación y prueba (tablas **4-22** y **4-23**, respectivamente), los errores de los modelos permanecen por debajo de los errores cometidos por SGP4. Incluso en el caso del Mod_2, cuyas predicciones, como se ha mencionado anteriormente, muestran un comportamiento inestable a partir del sexto día de propagación.

Por último, cabe señalar que la serie $\varepsilon_{212}^{\theta}$, que experimentó el mayor error en distancia durante la fase de entrenamiento del modelo secuencial Mod_1, no presenta los errores más

Modelo	Error en distancia km – Tiempo en días					
	2	4	6	8	10	12
SGP4	6,84	12,11	16,97	22,38	29,01	33,93
Mod_{Opt}	0,56	0,56	0,61	0,74	1,18	1,61
Mod_1	1,73	4,54	8,23	11,13	15,12	18,24
Mod_2	1,57	3,97	7,14	9,24	12,02	13,42

Tabla 4-21.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 80. El comienzo de las predicciones coincide con el inicio del periodo de entrenamiento.

Modelo	Error en distancia km – Tiempo en días					
	10	12	14	16	18	20
SGP4	29,01	35,56	39,73	45,39	49,22	52,71
Mod_{Opt}	1,27	1,61	1,83	2,03	2,03	2,05
Mod_1	1,23	1,62	2,31	4,97	7,41	10,32
Mod_2	1,48	3,81	9,12	14,15	21,66	32,38

Tabla 4-22.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 80. El comienzo de las predicciones coincide con el inicio del periodo de validación.

Modelo	Error en distancia km – Tiempo en días					
	12	14	16	18	20	22
SGP4	33,93	39,73	43,92	49,22	52,71	54,67
Mod_{Opt}	1,61	1,83	1,99	2,03	2,05	2,05
Mod_1	1,73	2,41	4,13	7,24	9,57	12,04
Mod_2	1,86	4,84	10,50	15,31	28,88	30,83

Tabla 4-23.: Errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN para el TLE 80. El comienzo de las predicciones coincide con el inicio del periodo de prueba.

altos en el caso aleatorio. Estos errores son 1.34, 3.58, 6.87, 9.44, 13.24 y 14.27 km en los días 2, 4, 6, 8, 10 y 12 de propagación, respectivamente. Por otro lado, los errores para la serie $\varepsilon^{\theta}_{80}$, que muestra el peor comportamiento en el caso aleatorio, son 1.73, 4.54, 8.22, 11.13, 15.12 y 18.23 km en los mismos instantes de tiempo.

Análisis de los modelos de RN. Conjunto de entrenamiento

En la figura **4-14** se presentan los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y los dos modelos de RN para un horizonte de propagación de 12 días. El número de valores atípicos se muestra en la tabla **4-24**, y como en los casos anteriores, estos datos no se incluyen en los diagramas. En cuanto a la diferencia entre el bigote superior e inferior en los días 2, 4, 6, 8, 10 y 12 de propagación, esta es de

aproximadamente 1.34, 3.87, 7.39, 8.91, 13.6 y 17.94 km, respectivamente, en el caso del Mod_2 que presenta los valores más altos. Comparando con los resultados en el caso máximo de los modelos secuenciales, se observa que la dispersión se reduce en 0.35, 0.34, 0.85, 3.94, 11.16 y 23.52 km, respectivamente. Los errores más grandes se observan en el Mod_2, con 1.87, 4.82, 8.72, 11.13, 15.31 y 20.84 km durante los 12 días de propagación. Por otro lado, los errores más pequeños los presenta el Mod_1, con 1.86, 4.59, 8.31, 11.13, 15.12 y 18.36 km. Este modelo, a su vez, muestra la menor dispersión. Comparado con los mínimos errores encontrados en los modelos secuenciales en la fase de entrenamiento, se observa que el error se redujo en 0.10 km a los 2 días de propagación, en 0.27 a los 10 días y en 9.64 km a los 12 días. Sin embargo, tuvo un leve incremento de 0.05, 0.99 y 0.66 km los días 4, 6 y 8, respectivamente. También se observa que durante los primeros 4 días de propagación, el error en las predicciones de los modelos no supera el cuartil Q_1 (7.35 km al día 4) de SGP4. Para los días restantes, los errores no sobrepasan su mediana, cuyo valor es de unos 22.34 km a los 12 días de propagación.

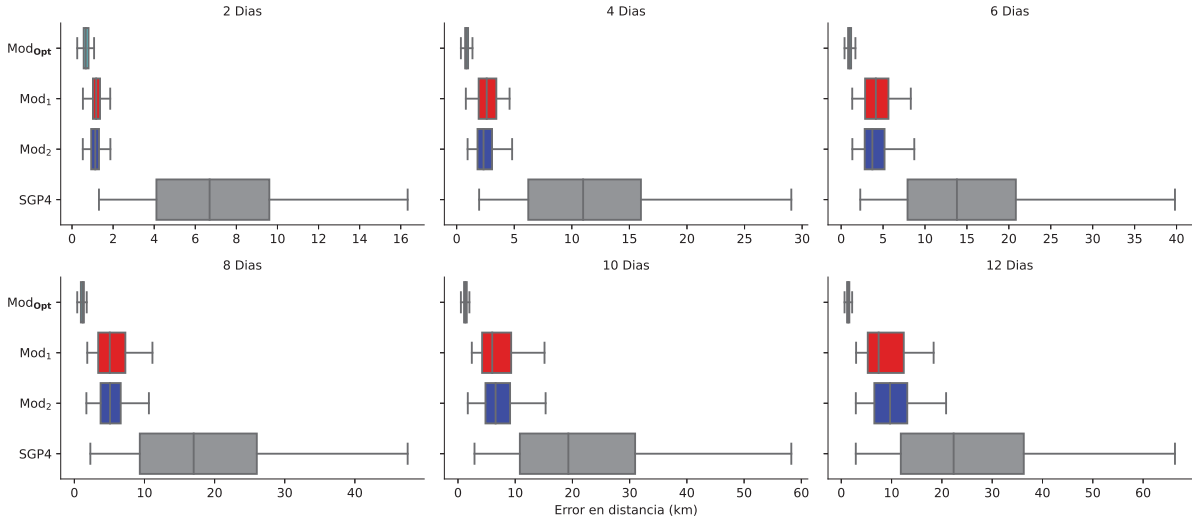


Figura 4-14.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los dos modelos de RN con los datos de la fase de entrenamiento.

La mayor cantidad de valores atípicos se observó durante los primeros 2 días de propagación. En este período, el modelo Mod_2 registró el número más alto con 8 valores atípicos, mientras que Mod_1 tuvo 7. En los días siguientes, Mod_1 no mostró valores atípicos, a diferencia del Mod_2, cuya cantidad disminuyó gradualmente hasta llegar a 3 valores en el día 12. Es importante destacar que el número de valores atípicos es mucho menor en comparación con el caso secuencial, como se puede apreciar en la tabla **4-11**.

En la figura **4-15** se muestra el diagrama de caja y bigotes del análisis de $\mathcal{I}_{m_{SGP4}}$ para un horizonte de propagación de 12 días utilizando los datos del conjunto de entrenamiento. Se observa que los modelos muestran una diferencia de aproximadamente 1.2 unidades entre el bigote superior e inferior. Sin embargo, la caja del modelo Mod_1 contiene los valores

Modelo	Número de valores atípicos - Tiempo en días					
	2	4	6	8	10	12
SGP4	0	0	0	1	1	1
Mod_{Opt}	3	8	0	0	0	0
Mod_1	7	0	0	0	0	0
Mod_2	8	6	1	2	1	3

Tabla 4-24.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-14** según el día de propagación.

más pequeños. Se han identificado un total de 28 valores atípicos en ambos modelos, de los cuales 10 corresponden al Mod_1 y 18 al Mod_2. Estos valores atípicos provienen de 20 series, de las cuales 13 están clasificadas como sin tendencia, 6 como de tendencia positiva y 1 de tendencia negativa. Es importante destacar que todas las series presentes en el Mod_1 también se encuentran en Mod_2. Además, las 13 series sin tendencia son las mismas que generaron valores atípicos en los modelos secuenciales. En términos generales, el número de casos en que las predicciones de los modelos empeoraron las de SGP4 a 2, 4, 6, 8 10 y 12 días de propagación fueron de 0, 4, 10, 15, 19 y 22 para Mod_1 y de 0, 7, 12, 17, 19 y 24 para Mod_2, respectivamente. El indice de mejora mas alto mostrado por el Mod_1 es de 5.34 unidades (un valor atípico) y corresponde a la serie sin tendencia $\varepsilon_{31}^{\theta}$, donde el error se incrementó de 4.47 a 23.86 km.

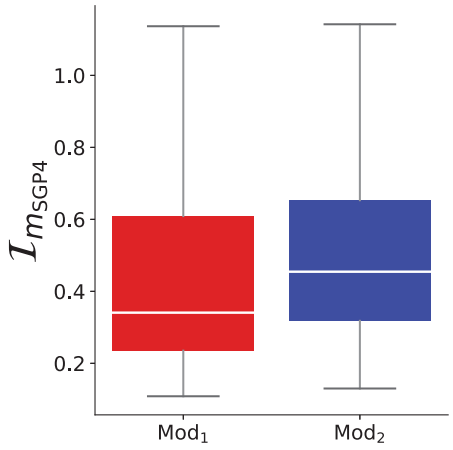


Figura 4-15.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto a SGP4 para los modelos Mod_1 y Mod_2, para un horizonte de propagación de 12 días con los datos de entrenamiento.

Análisis de los modelos de RN. Conjunto de validación

En la figura **4-16** se presentan los diagramas de caja y bigotes que analizan los máximos del error en distancia entre AIDA y SGP4, Mod_{Opt}, y los dos modelos de RN para cada uno de

los 219 TLE a partir del décimo día de propagación. La tabla **4-25** proporciona información sobre el número de valores atípicos, los cuales no se representan en los diagramas, siguiendo la misma práctica adoptada en casos anteriores.

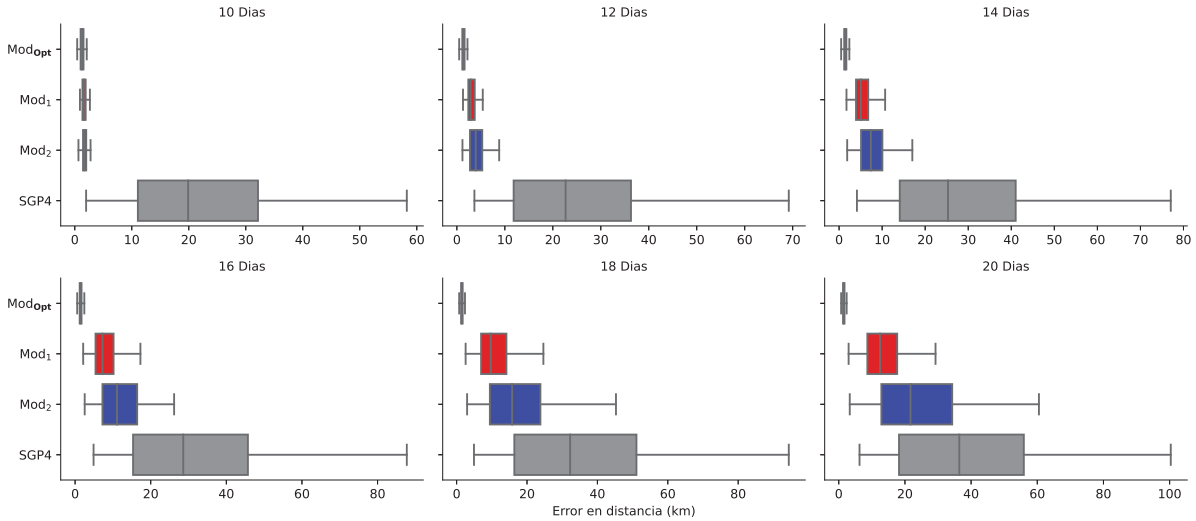


Figura 4-16.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\mathrm{Opt}}$ y los dos modelos de RN con los datos de la fase de validación.

Modelo	Número de valores atípicos - Tiempo en días					
	10	12	14	16	18	20
SGP4	1	1	1	1	1	1
Mod$_{\mathrm{Opt}}$	0	1	0	0	5	4
Mod$_1$	10	17	15	13	2	2
Mod$_2$	6	15	18	11	22	24

Tabla 4-25.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-16** según el día de propagación.

A lo largo del período de propagación, la diferencia entre el bigote superior e inferior del Mod$_2$, el cual presenta la mayor dispersión, es de aproximadamente 2.22, 7.65, 15.13, 23.65, 42.3 y 57.18 km. Esto reduce la dispersión de SGP4 en aproximadamente 54.04, 57.88, 57.77, 59.31, 47.19 y 36.89 km, respectivamente. Durante el período de propagación, los errores más altos se observan en Mod$_2$, con 2.86, 8.82, 17.01, 26.19, 45.28 y 60.5 km. Se destaca que el máximo error del Mod$_2$ supera tanto el primer cuartil (14.25 km) como el tercer cuartil (56.44 km) de los errores de SGP4, los días 14 y 20 de propagación respectivamente. Por otro lado, los mínimos errores se presentan en el Mod$_1$ con 2.65, 5.40, 10.70, 17.56, 24.65 y 31.25 km. El máximo error del Mod$_1$ no sobrepasa la mediana del error de SGP4 a lo largo de todo el período de propagación. Comparando con los mínimos errores de los modelos secuenciales en la fase de validación se evidencia una reducción del error más pequeño de 3.08, 16.95,

36.77, 50.67, 57.36 y 69.46 km, respectivamente. Es importante notar que los errores más pequeños se obtuvieron con un solo modelo, mientras en el caso secuencial, esos errores se obtuvieron con los modelos Mod_1 y Mod_2. La mayor concentración de valores atípicos se presentó en el día 14 de propagación, donde se registraron 33 valores. De estos valores, 18 fueron generados por Mod_2. En el día 20 de propagación, Mod_1 registró únicamente 2 valores atípicos, mientras que Mod_2 registró 24.

En la figura **4-17** se muestra el diagrama de caja y bigotes con el análisis de $\mathcal{I}_{m_{SGP4}}$ para un horizonte de propagación de 12 días utilizando los datos del conjunto de validación. La diferencia entre el bigote superior e inferior es de aproximadamente 0.8 unidades para Mod_1 y aproximadamente 1.3 unidades para Mod_2. Según estos resultados, en ambos casos los modelos logran mejorar las predicciones de SGP4 en más del 75 % de los casos, destacándose el modelo Mod_1 por presentar resultados más precisos. En total, se han identificado 34 valores atípicos, de los cuales 30 pertenecen al Mod_1 y 4 al Mod_2, respectivamente. El indice de mejora mas alto mostrado por el Mod_1, fue de 3.17 y se registró en la serie $\varepsilon^{\theta}_{206}$, donde el error aumentó de 6.33 a 20.08 km. Los casos en los cuales las predicciones del Mod_1 empeoraron en comparación con las de SGP4 a 10, 12, 14, 16, 18 y 20 días de propagación, fueron de 0, 2, 7, 23, 27, 22 y 27. Mientras que para el modelo Mod_2, fueron 0, 1, 8, 16, 16, 20 y 40, respectivamente.

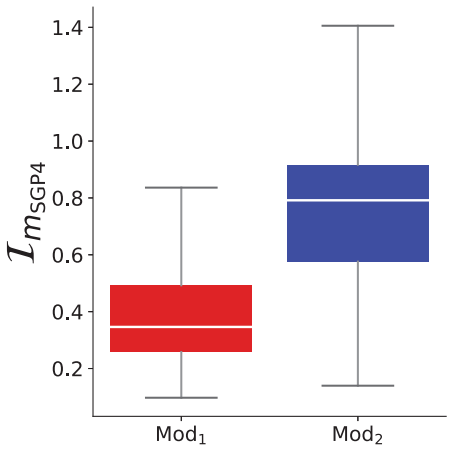


Figura 4-17.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto a SGP4 para los modelos Mod_1 y Mod_2, para un horizonte de propagación de 12 días con los datos de validación.

Análisis de los modelos de RN. Conjunto de prueba

Los diagramas de caja y bigotes que analiza los valores máximos del error en distancia entre AIDA y SGP4, Mod_{Opt}, y los dos modelos de RN se presentan en la figura **4-18**. Este análisis se lleva a cabo para cada uno de los 219 TLE a partir del duodécimo día de propagación. La Tabla **4-26** muestra el número de valores atípicos encontrados, los cuales no se incluyen en los diagramas.

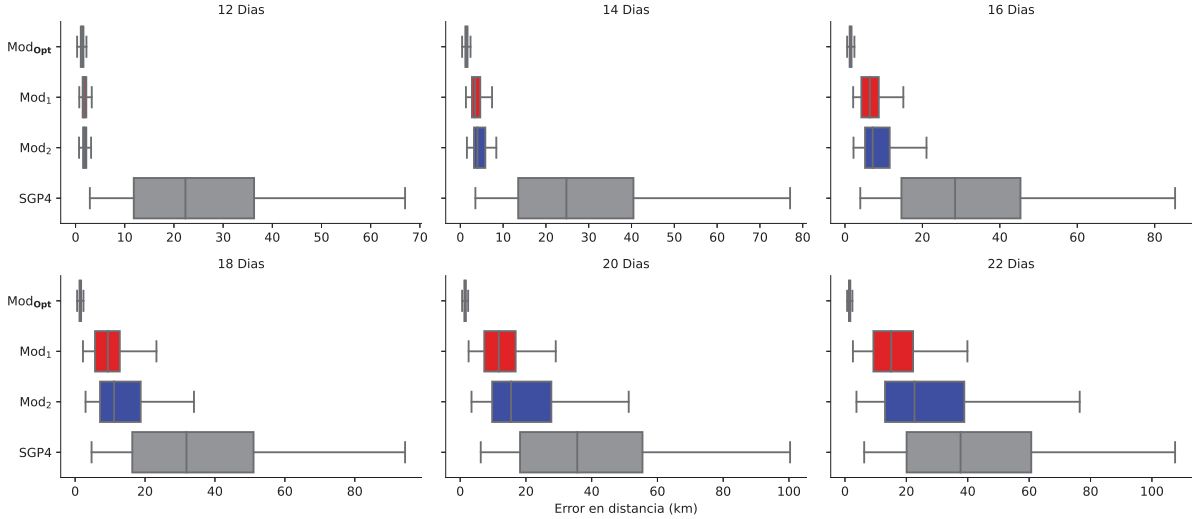


Figura 4-18.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y los dos modelos de RN con los datos de la fase de prueba.

Modelo	Número de valores atípicos - Tiempo en días					
	12	14	16	18	20	22
SGP4	1	1	1	1	1	1
Mod$_{\text{Opt}}$	0	1	0	3	1	2
Mod$_1$	17	28	15	2	2	1
Mod$_2$	16	31	11	21	30	33

Tabla 4-26.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-18** según el día de propagación.

La diferencia entre el bigote superior e inferior los días 12, 14, 16, 18, 20 y 22 de propagación, es de aproximadamente 2.45, 6.87, 18.85, 31.01, 47.82 y 75.28 km para Mod$_2$ que presenta la mayor dispersión. Esto indica una reducción en la dispersión frente a SGP4, de aproximadamente 53.81, 58.66, 54.05, 51.95, 41.67 y 18.79 km, respectivamente. Si se compara con el modelo óptimo, la diferencia mas pequeña, unos 0.55 km, se observa a los primeros 2 días de propagación (día 12). Los errores más altos durante todo el periodo, se presentan con este modelo, unos 3.29, 8.41, 21.06, 34.00, 51.27 y 79.03 km. No obstante, a diferencia de los modelos secuenciales, estos errores no sobrepasan los errores cometidos por SGP4. Los mínimos errores se observan en el Mod$_1$ con 3.29, 7.64, 15.58, 23.26, 29.07 y 39.89 km. En comparación con los mínimos errores mostrados de la fase de prueba de los modelos secuenciales, se observa una reducción de 6.63, 24.22, 48.31, 67.38, 80.67 y 86.95 km, respectivamente. Al igual que en la fase de validación, los mínimos errores se obtienen con un solo modelo, mientras que en los modelos secuenciales, los mínimos errores se obtienen combinando los modelos Mod$_1$ y Mod$_2$.

El número de valores atípicos muestra un aumento durante los primeros días de propaga-

ción en comparación con la fase de validación. Este aumento es notable en Mod_2, que pasó de tener 6 casos en el conjunto de validación a 16 en el conjunto de prueba durante los primeros 2 días de propagación, y de 24 a 33 al término de los 12 días. Sin embargo, se observa una reducción en los valores atípicos para Mod_1, llegando a presentar un único caso el día 22.

El diagrama de caja y bigotes con el análisis de $\mathcal{I}_{m_{SGP4}}$ para un horizonte de propagación de 12 días con los datos del conjunto de prueba se muestra en la figura **4-19**. La diferencia entre el bigote superior e inferior es de aproximadamente 0.9 unidades para Mod_1 y aproximadamente 1.4 unidades para Mod_2. Entre ambos modelos se tienen un total de 30 valores atípicos, de los cuales 26 pertenecen a Mod_1. En este modelo se observa el valor atípico más alto, unas 2.22 unidades, que está asociado a la serie $\varepsilon^{\theta}_{206}$, donde el error crece de 6.73 a 14.94 km. El número de casos en los que las predicciones de ambos modelos resultaron ser peores que las de SGP4 a 12, 14, 16, 18, 20 y 22 días fueron de 0, 0, 7, 16, 19 y 23 para Mod_1, y de 3, 4, 10, 14, 22 y 52 casos para Mod_2. Al observar la gráfica, se puede comprobar que el cuartil Q_3 en ambos modelos es inferior a la unidad, lo cual indica que más del 75 % de los casos mejoran con respecto a SGP4. Nuevamente, destacan los resultados obtenidos por Mod_1 en comparación con los obtenidos por Mod_2.

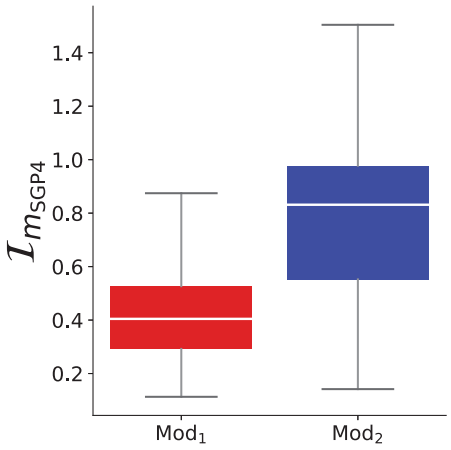


Figura 4-19.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto a SGP4 para los modelos Mod_1 y Mod_2, para un horizonte de propagación de 12 días con los datos de prueba.

Selección del mejor modelo

La selección del modelo con el mejor rendimiento resulta más sencilla que en el caso de los modelos secuenciales. En estos últimos se tuvo en cuenta que algunos modelos mostraban un mejor comportamiento a corto plazo, mientras que otros lo hacían a largo plazo. Sin embargo, en el caso de los modelos aleatorios, el Mod_1 muestra un mejor desempeño en ambas situaciones. Según lo observado Mod_1 presenta la menor diferencia entre el bigote superior e inferior durante las fases de entrenamiento, validación y prueba. En los análisis del índice de mejora respecto a SGP4, los valores en los gráficos de caja y bigotes del Mod_1

son más pequeños que los respectivos valores del Mod_2, lo que indica una mayor reducción en el error en distancia frente a su competidor. Además, este modelo presentó el menor número de valores atípicos durante los tres periodos. Durante el entrenamiento, Mod_1 presentó un total de 7 valores atípicos en los 12 días de propagación, mientras que Mod_2 mostró 21. En la validación, Mod_1 registró un total de 59 valores atípicos, en comparación con los 96 del Mod_2. En la fase de prueba, se contabilizaron 65 valores atípicos en total, menos de la mitad de los 142 presentados por Mod_2.

Por último, se va a comparar la calidad de las predicciones de cada uno de los modelos empleando el índice de mejora respecto al óptimo. En la figura **4-20** se muestra el diagrama de caja y bigotes con el análisis de $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para los dos modelos de RN y un horizonte de propagación de 12 días en los conjuntos de entrenamiento, validación y prueba (mostrados de izquierda a derecha, respectivamente). En este índice, cuanto más cercanas estén las predicciones de los modelos basados en RN a las predicciones del modelo óptimo, más cercano es el valor del índice óptimo a 1. Como se puede observar, en los tres casos, los resultados obtenidos por Mod_1 se acercan más al valor óptimo que Mod_2.

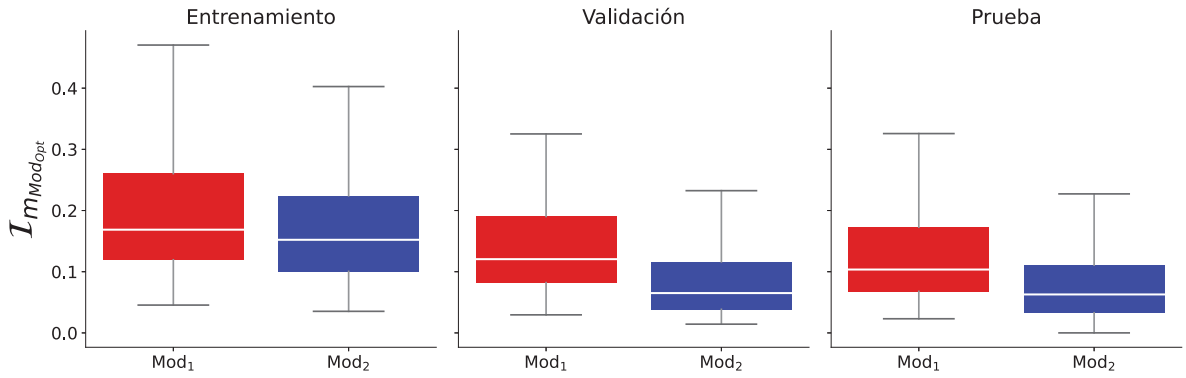


Figura 4-20.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto al óptimo $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para los dos modelos aleatorios con los datos de entrenamiento, validación y prueba. El horizonte de propagación es de 12 días.

A lo largo de este trabajo, los modelos aleatorios demostraron ser más robustos que los secuenciales, ya que sus predicciones mejoraron notablemente los resultados de SGP4, independientemente de si se aplicaron en los conjuntos de entrenamiento, validación o prueba. No obstante, la ordenación aleatoria de los datos dificulta la convergencia durante el proceso de entrenamiento de la RN, lo que hace más difícil encontrar un modelo de red neuronal con un buen ajuste a los datos.

Finalmente, es importante destacar que el comportamiento de SGP4 es bueno para algunos TLE que generan series sin tendencia, lo que reduce significativamente el margen de mejora del propagador híbrido. En estos casos, los errores de SGP4 a 12, 20 y 22 días son de 15.78, 16.2 y 16.48 km, respectivamente. Estos resultados son altamente satisfactorios, especialmente al considerar que en casos de series con tendencia negativa, los errores pueden alcanzar los 21, 70 y 74 km para esos mismos días, mientras que en las series con tendencia

positiva, los errores pueden ascender hasta 62, 123 y 132 km, respectivamente, para los TLE estudiados.

4.2.5. Robustez del Mod_1

Este modelo ha sido entrenado con 219 series de las 312 disponibles, las cuales fueron obtenidas a partir de los TLE del satélite GSAT0203. Las 93 series restantes serán utilizadas para probar la robustez del Mod_1. Este análisis evalúa la capacidad del modelo para reproducir el comportamiento de las series desde el instante inicial o desde cualquier otro instante. En este estudio, se considerarán dos instantes: los días 8 y 10, que corresponden con los inicios de los conjuntos de validación y prueba utilizados durante el entrenamiento del Mod_1, respectivamente.

Utilizando el instante inicial se evalúa la capacidad del modelo para detectar patrones similares a los procesados durante la etapa de entrenamiento del Mod_1. A partir del día 8, las predicciones del modelo se basan en parte en el reconocimiento de patrones similares a los aprendidos y en parte a su capacidad para predecir patrones no conocidos. Por último, a partir del día 10, se examina la capacidad del modelo para predecir exclusivamente patrones no conocidos, ya que no cuenta con ninguna información sobre los que pueden aparecer a partir de este día en las series con las que se entrenó el modelo.

En este análisis también se utilizará el error en la distancia entre AIDA y $\text{HEncke}_{\mathbf{SGP4}}$ para evaluar el Mod_1. El horizonte de propagación considerado es de 12 días y las predicciones se inician en los días 2, 10 y 12. Los errores se muestran en los días 2, 4, 6, 8, 10 y 12 posteriores al inicio de las predicciones.

Análisis a partir del día 2

En la figura **4-21** se presentan los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y Mod_1 para un horizonte de propagación de 12 días. En este caso, no se detectaron valores atípicos.

La diferencia entre el bigote superior y el inferior para SGP4 a los 2, 4, 6, 8, 10 y 12 días de propagación es de aproximadamente 11.64, 20.00, 27.67, 36.41, 47.21 y 54.35 km, respectivamente. El modelo Mod_1 es capaz de reducir esta diferencia a aproximadamente 1.14, 3.86, 6.69, 9.14, 12.27 y 15.84 km. El máximo error alcanzado por SGP4 para estos días es de 13.2, 22.33, 30.32, 39.07, 49.96 y 58.27 km. Por su parte, los máximos errores cometidos por Mod_1 están alrededor de los 1.79, 4.59, 8.31, 10.91, 14.91 y 18.49 km. Los resultados del Mod_1 están notablemente más cercanos que los de SGP4 a los 1.11, 1.33, 1.72, 1.79, 1.79 y 1.98 km obtenidos con el modelo óptimo. Durante los primeros 6 días de propagación, el error en la predicción del Mod_1 no supera el cuartil Q_1 de SGP4, que es de 9.04 km al día 6. Para los días restantes, la magnitud del error no sobrepasa la mediana de SGP4, que es de unos 22.76 km al día 12.

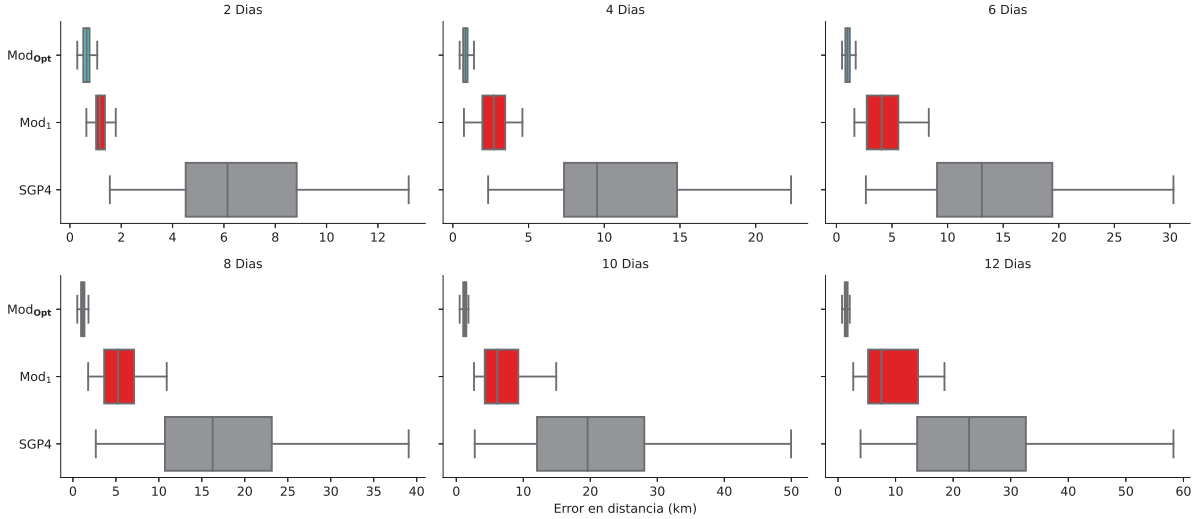


Figura 4-21.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\mathrm{Opt}}$ y Mod$_1$ de las 93 series. Las predicciones inician a partir del día 2.

Análisis a partir del día 10

La figura **4-22** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod$_{\mathrm{Opt}}$, y Mod$_1$ para cada uno de los 93 TLE a partir del décimo día de propagación.

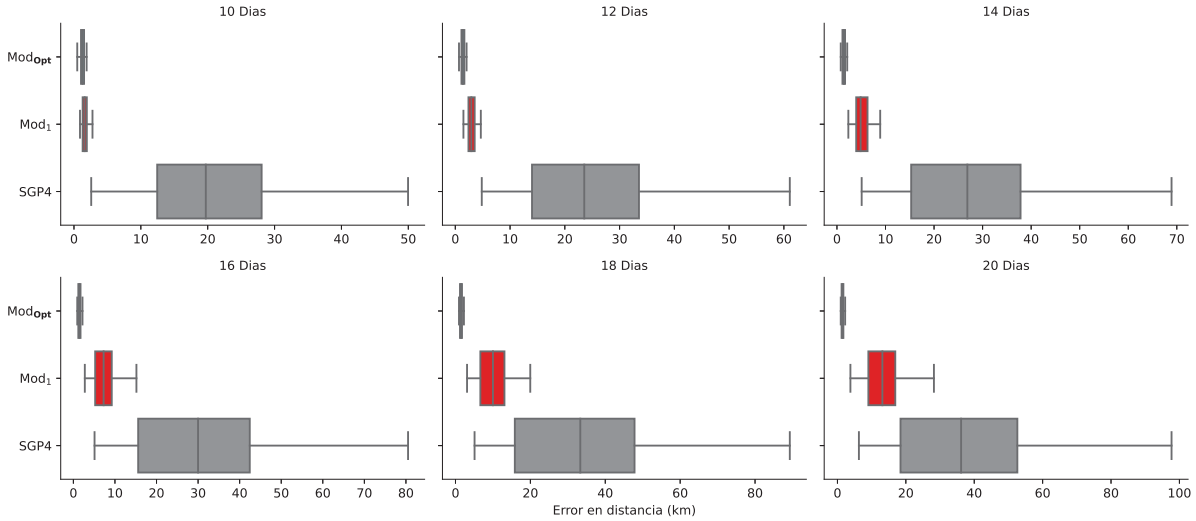


Figura 4-22.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\mathrm{Opt}}$ y Mod$_1$ en las 93 series. Las predicciones inician a partir del día 10.

Durante los días del 10 al 20 de propagación, se observa una diferencia significativa entre los valores de los bigotes superior e inferior para el modelo SGP4. Estas diferencias se sitúan en torno a los 47.38, 56.25, 63.82, 75.42, 84.19 y 91.42 km, respectivamente. Por otro lado, en el caso del modelo Mod_1, se observan diferencias de 1.87, 3.18, 6.56, 12.41, 16.88 y 24.42 km durante el mismo intervalo de tiempo. Esto indica una reducción en la dispersión del error a los 12 días de aproximadamente 63.21 kilómetros. En cuanto al modelo óptimo, las diferencias entre los bigotes son de 1.40, 1.38, 1.33, 1.26, 1.32 y 1.29 km, respectivamente. La menor diferencia con respecto al Mod_1 se observa el día 10 de propagación, siendo de 0.47 kilómetros. Los máximos errores cometidos por este modelo son de aproximadamente 2.77, 4.66, 8.91, 15.17, 19.97 y 28.21 km. El máximo error alcanzado por SGP4 es de 49.96, 61.08, 68.92, 80.51, 89.28 y 97.7 km, respectivamente, lo que representa una reducción de este error de aproximadamente 47.19, 56.42, 60.01, 65.34, 69.31 y 69.49 km. Hasta el día 18, los errores cometidos por Mod_1 se mantienen por debajo del cuartil Q_3 de SGP4, que es de 15.58 km. En las propagaciones posteriores, el máximo del modelo entrenado no supera la mediana del modelo SGP4, que se sitúa alrededor de los 36.15 kilómetros el día 20.

Es importante mencionar que el error mínimo cometido por SGP4 en el día 10 es de aproximadamente 2.57 km, mientras que el cuartil Q_3 del Mod_1 se sitúa en torno a los 1.92 km. Esto indica que más del 75 % de los errores del modelo basado en RN se encuentra por debajo del mínimo error cometido por SGP4.

Durante los 12 días de propagación, se registraron un total de 24 valores atípicos distribuidos como sigue: 3, 6, 6, 5, 3 y 1 para los días 10, 12, 14, 16, 18 y 20, respectivamente. Se observa que la mitad de estos valores ocurrieron en los días 12 y 14, mientras que solo se presentó un caso en el día 20. En este caso, tanto SGP4 como Mod_{Opt} no presentaron valores atípicos.

Análisis a partir del día 12

La figura **4-23** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y Mod_1 para cada uno de los 93 TLE a partir del día 12 de propagación.

La diferencia entre el bigote superior e inferior para SGP4 durante el período de propagación de los días 12 a 22 es de aproximadamente 54.74, 63.82, 72.60, 84.19, 91.03 y 98.15 km, respectivamente. En el caso del Mod_1, estas diferencias son de aproximadamente 1.99, 4.76, 9.59, 14.56, 18.99 y 24.91 km en el mismo intervalo de tiempo. Esto indica una reducción de la dispersión del error en el día 22 de aproximadamente 73.24 km, ligeramente mayor que en el caso anterior. En cuanto a Mod_{Opt}, presenta una diferencia de alrededor de 1.54, 1.40, 1.30, 1.34, 1.34 y 1.34 km, respectivamente. Es decir, durante los primeros 2 días de propagación (al día 12), la diferencia con Mod_1 es de 0.45 km. Comparando con la diferencia de 0.47 km obtenida en el caso anterior, se evidencia una ligera mejora de aproximadamente 0.02 km. Los mayores errores que presenta Mod_1, son de aproximadamente 2.99, 6.05, 11.55, 17.25, 22.18 y 28.5 km. En este mismo periodo, los mayores errores de SGP4 son de 58.88, 68.91, 77.69, 89.28, 97.30 y 104.77 km, lo que implica una reducción en el máximo error en distancia de 55.89, 62.86, 66.13, 72.03, 75.12 y 76.27 km, respectivamente.

En las propagaciones realizadas hasta el día 16, los errores del Mod_1 no superan los 15.59

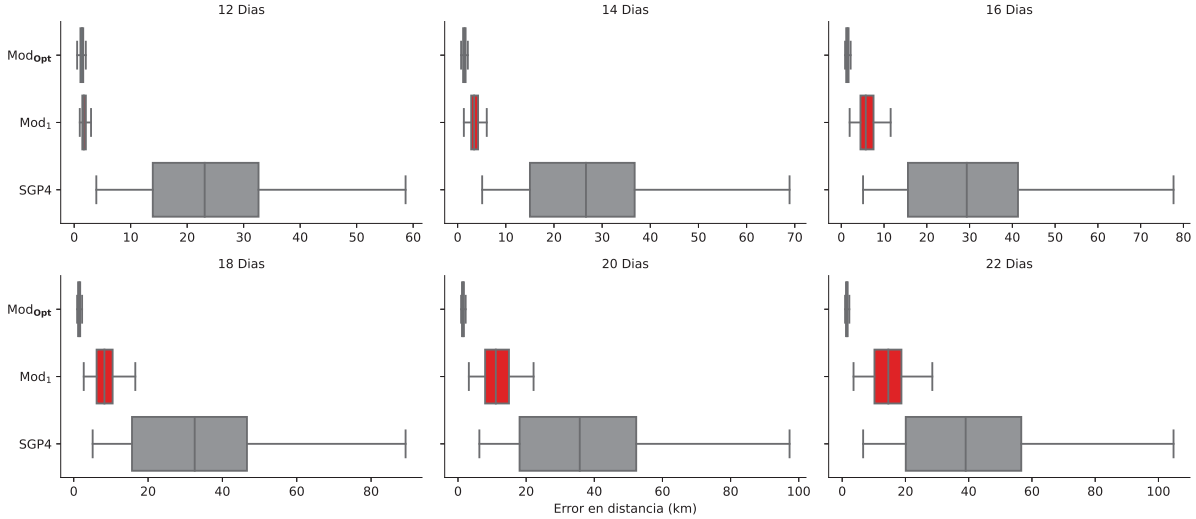


Figura 4-23.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y Mod$_1$ en las 93 series. Las predicciones se inician a partir del día 12.

km, que es el valor en el que se encuentra el cuartil Q_1 de SGP4 ese día. Se destaca el hecho de que para el día 12, el máximo error cometido por Mod$_1$ fue de 3.00 km, que no supera los 3.91 km del mínimo error de SGP4. En este mismo día, el máximo error alcanzado por Mod$_{\text{Opt}}$ fue de aproximadamente 2.59 km, lo que indica una diferencia de menos de un kilómetro. Al llegar al día 22 de propagación, el Q_3 del Mod$_1$ (18.73 km) no sobrepasa el Q_1 de SGP4 (20.10 km).

Durante el periodo de propagación, se registraron un total de 33 valores atípicos, distribuidos de la siguiente manera: 4, 8, 5, 6, 5 y 5 para los días 12, 14, 16, 18, 20 y 22, respectivamente. Al igual que en los casos anteriores, SGP4 y Mod$_{\text{Opt}}$ no presentaron valores atípicos.

Índices de mejora

En la tabla **4-27** se muestra el número de casos en los que las predicciones del modelo Mod$_1$ empeoraron en comparación con SGP4.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días						
	Días	2	4	6	8	10	12
Mod$_1$	0	0	1	6	6	6	6
	8	0	1	1	7	8	11
	10	0	0	2	7	8	10

Tabla 4-27.: Números de casos en los que las predicciones del modelos Mod$_1$ empeoraron las de SGP4 desde el instante inicial, así como desde los días 8 y 10.

Durante los primeros 2 días de propagación, no se registró ningún caso en el que las

predicciones del modelo fueran peores que las de SGP4. En el día 4, solo se presentó un caso. A partir del día 6, se presentaron 6 casos, cifra que se mantuvo hasta el día 12. Para los días 2, 4, 6, 8, 10 y 12 se observaron 3, 4, 7, 5, 6 y 5 valores atípicos, respectivamente. El caso más notable ocurrió el día 10, con un valor atípico de 2.80 unidades, correspondiente a la serie sin tendencia $\varepsilon^{\theta}_{61}$, donde el error aumentó de 5.59 a 15.65 km. En las propagaciones realizadas a partir del día 8, se observó el primer caso donde las predicciones del Mod_1 empeoraron en comparación con SGP4, el cual se presentó en el día 12. Estos casos aumentaron gradualmente hasta alcanzar 11 el día 20. Durante este período, se presentaron 11, 3, 6, 8, 11 y 11 valores atípicos, respectivamente. El valor atípico más alto fue de 3.17 unidades el día 20, asociado a la serie sin tendencia $\varepsilon^{\theta}_{207}$, donde el error aumentó de 5.68 a 18.05 km. En el último período, se registraron 2 casos el día 16 donde Mod_1 empeoró los resultados de SGP4, y a partir de este día, los casos aumentaron hasta llegar a 10 el día 22. El valor atípico más alto fue de 2.22 unidades, nuevamente relacionado con la serie $\varepsilon^{\theta}_{207}$, donde el error de la predicción pasó de 6.73 a 14.94 km.

En la figura **4-24** se presentan los diagramas de caja y bigotes que analizan $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para el modelo Mod_1 durante una propagación de 12 días, comenzando desde los días 2, 8 y 10 (mostrados de izquierda a derecha, respectivamente). La diferencia entre el bigote superior e inferior en los tres casos es de 0.39, 0.23 y 0.28 unidades, respectivamente. En las propagaciones desde el día 2, no se registraron valores atípicos. Sin embargo, en las propagaciones desde el día 10 se presentaron 2 valores atípicos, y en el último caso se observaron 6. Es importante destacar que, si bien el modelo muestra buenos resultados, aún existe un amplio margen para mejoras.

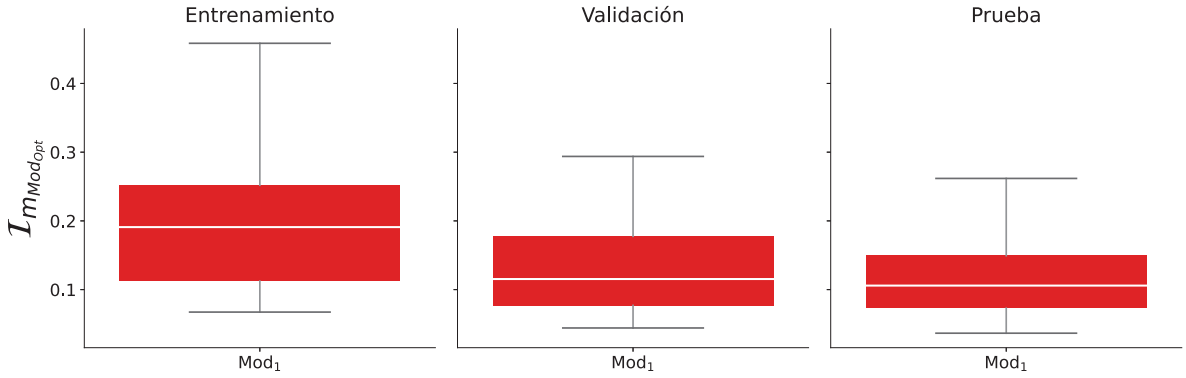


Figura 4-24.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto al óptimo $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para Mod_1 aleatorio en las 93 series. El horizonte de propagación es de 12 días.

4.2.6. Generalización del Mod_1

Finalmente, se evalúa la capacidad de generalización del Mod_1 utilizando las 1685 series generadas a partir de los datos del satélite GALILEO-FM3. Es importante recordar que estas series muestran un comportamiento similar a las series de entrenamiento, pero también

presentan patrones que no han sido observados por las redes neuronales. El proceso de evaluación es análogo al descrito en las pruebas de robustez. Al igual que en los casos anteriores, el horizonte de propagación es de 12 días, y los errores se registran a los 2, 4, 6, 8, 10 y 12 días posteriores al inicio de las predicciones.

Análisis a partir del día 2

En la figura **4-25** se muestran los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y Mod_1. La diferencia entre los bigotes superior e inferior para el modelo SGP4 a los 2, 4, 6, 8, 10 y 12 días es de 18.69, 34.06, 46.84, 62.80, 75.44 y 87.92 km, respectivamente. Por otro lado, Mod_1 reduce esta distancia a 1.98, 4.62, 7.80, 11.34, 15.74 y 20.58 km, mostrando una mejora significativa. Sin embargo, estos resultados aún están lejos del modelo óptimo, cuyas distancias se sitúan alrededor de 0.95, 1.04, 1.13, 1.38, 1.50 y 1.56 km. Los máximos errores cometidos por Mod_1 están alrededor de 2.43, 5.41, 9.05, 13.07, 17.8 y 22.78 km. mientras que los máximos errores de SGP4 son de 19.59, 35.78, 48.78, 64.87, 77.84 y 90.55 km, lo que implica una reducción de este error en 17.16, 30.37, 39.73, 51.8, 60.04 y 67.77 km, respectivamente. A lo largo del período de propagación, el cuartil Q_3 del Mod_1 no supera al Q_1 de SGP4. Por ejemplo, para el día 12, el Q_1 de SGP4 alcanza los 13.07 km, mientras que el Q_3 del Mod_1 es de 12.54 km. Además, en este mismo día, el máximo error del Mod_1 (22.78 km) no supera la mediana de SGP4 (27.95 km).

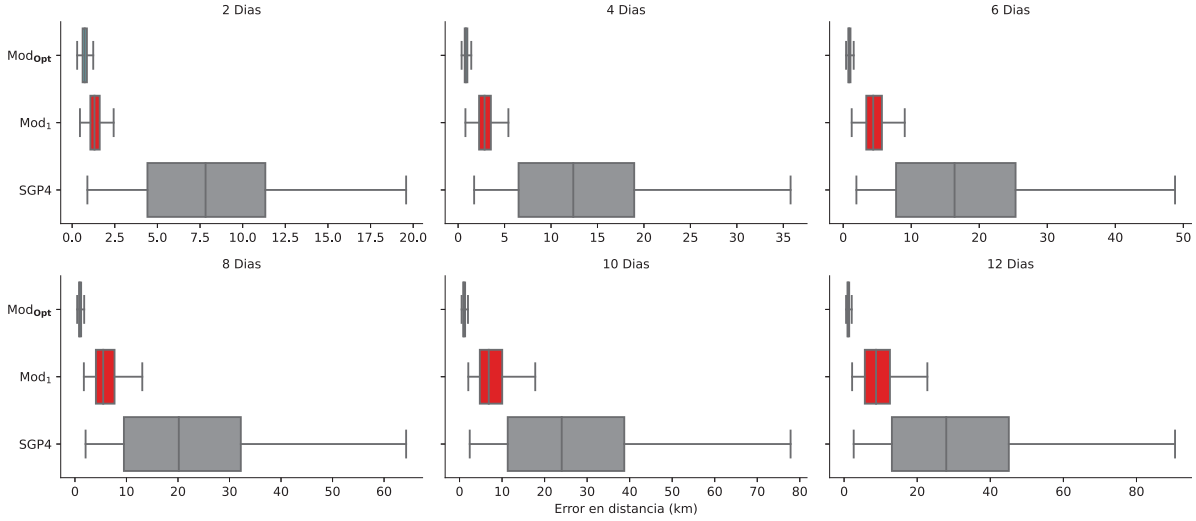


Figura 4-25.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y Mod_1 para las 1685 series. Las predicciones comienzan a partir del día 2.

Durante el período de propagación, el modelo Mod_1 mostró 48, 38, 20, 9, 7 y 5 valores atípicos para los días 2, 4, 6, 8, 10 y 12, respectivamente. El mayor número se registró en los primeros 2 días de propagación, con 48 casos. Por otro lado, el modelo óptimo exhibió

18, 57, 72, 36, 13 y 3 valores atípicos para los mismos días, con el mayor número de casos (72) observado a los 6 días de propagación. Es importante destacar que el modelo SGP4 no presentó valores atípicos en ningún momento.

Análisis a partir del día 10

La figura **4-26** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y Mod_1 para cada uno de los 1685 TLE a partir del día 10.

La diferencia entre el bigote superior e inferior de SGP4 al día 10 es de aproximadamente 78.87 km, mientras que para Mod_1 es de aproximadamente 2.49 km, situándose a solo 0.79 km del valor óptimo. Los máximos errores cometidos por Mod_1 en este periodo son de aproximadamente 2.99, 7.84, 12.62, 18.68, 26.07 y 36.32 km. Por otro lado, los máximos errores cometidos por SGP4 son de 79.92, 94.94, 107.77, 122.64, 137.53 y 151.17 km, lo que implica una reducción de aproximadamente 76.93, 87.1, 95.15, 103.96, 111.46 y 114.85 km, respectivamente.

A los 12 días de propagación (día 20), se observa que el máximo error del modelo Mod_1 no supera la mediana de SGP4, que se sitúa en aproximadamente 43.22 km.

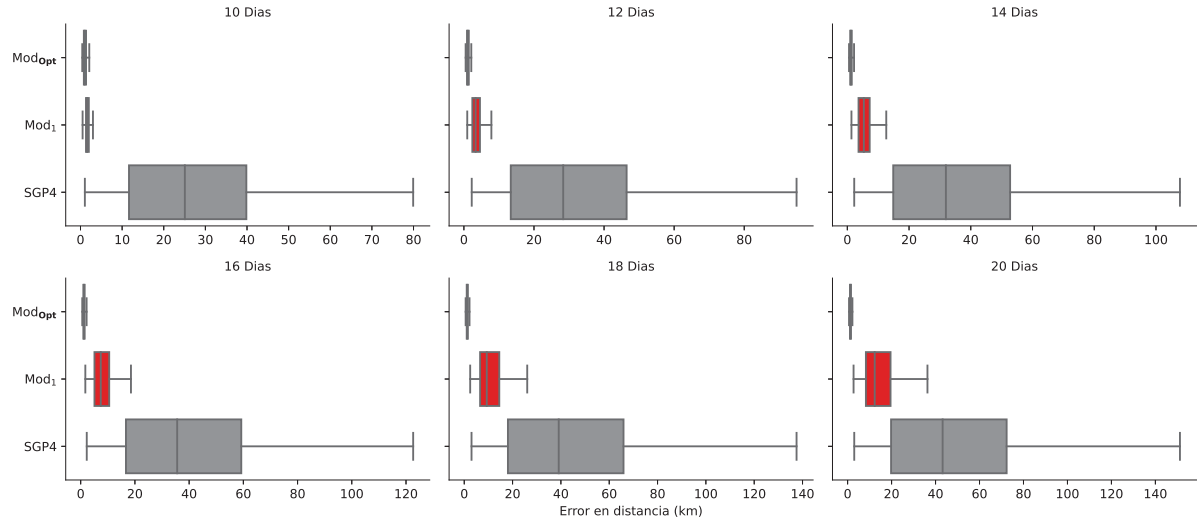


Figura 4-26.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y Mod_1 para las 1685 series. Las predicciones comienzan a partir del día 10.

A lo largo de los 12 días de propagación, Mod_1 registró 91, 63, 60, 91, 100 y 91 valores atípicos, alcanzando su máximo de 100 casos el día 10. Aunque este valor apenas difiere del promedio diario de aproximadamente 83 valores atípicos. Por otro lado, el modelo óptimo solo registró 2 valores atípicos, uno el día 10 y otro el día 12. En cuanto a SGP4, no se registraron valores atípicos.

Análisis a partir del día 12

La figura **4-26** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt}, y Mod_1 para cada uno de los 1685 TLE a partir del día 12.

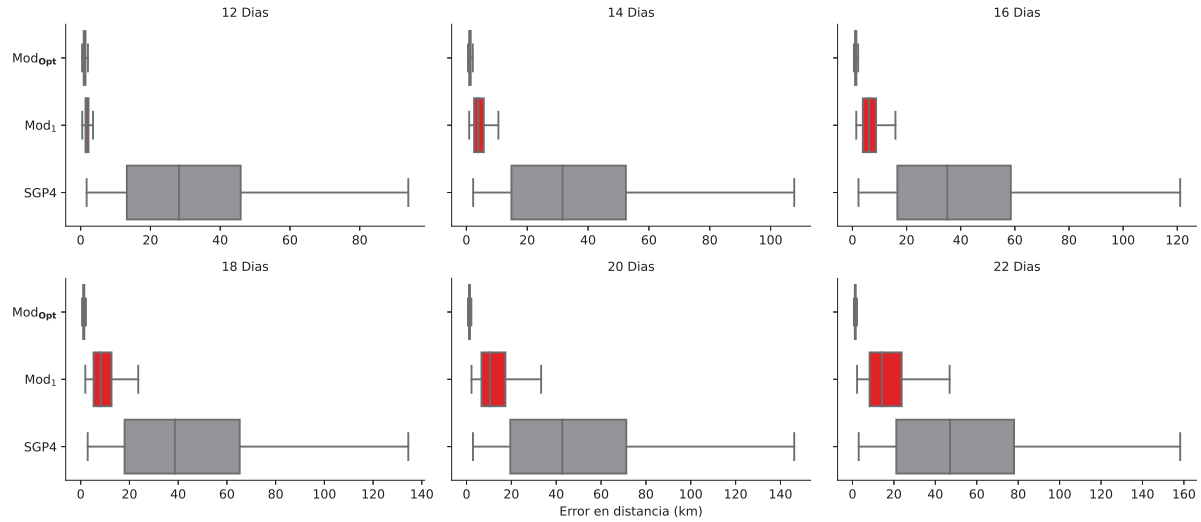


Figura 4-27.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y Mod_1 para las 1685 series. Las predicciones comienzan a partir del día 12.

La diferencia entre el bigote superior e inferior para Mod_1 es de aproximadamente 3.08 km, durante los primeros 2 días (día 12). Por su parte, SGP4 presenta una diferencia para este día de 92.2 km, lo que implica una reducción de la dispersión de SGP4 en al menos 89.12 km. En este mismo periodo, el modelo óptimo presentó un error de 1.54 km. Los máximos errores cometidos por Mod_1 son de aproximadamente 3.08, 9.59, 14.47, 21.71, 31.05 y 44.67 km. Por otro lado, los máximos errores alcanzados por SGP4 fueron de 93.92, 107.77, 121.08, 134.45, 146.14 y 158.2 km, lo que representa una reducción de 90.38, 97.25, 105.2, 110.84, 112.86 y 111.31 km, respectivamente. Es importante destacar que el máximo error del Mod_1 no supera la mediana de los errores cometidos por SGP4 al día 22, que se sitúa alrededor de los 47.04 km.

Se registraron 186, 72, 89, 107, 97 y 89 valores atípicos entre los días 12 y 22 de propagación. El mayor número de valores atípicos (186) se concentró durante los primeros 2 días. Por otro lado, para SGP4 solo se registraron 3 valores atípicos, los días 18, 20 y 22 de propagación. En este mismo periodo, Mod_{Opt} no registró valores atípicos.

Índices de mejora

En la tabla **4-28** se muestra el número de casos en los que las predicciones del Mod_1 empeoraron los resultados obtenidos con SGP4 en las 1685 series. Durante los primeros 2 días de propagación, se registraron 4 casos, aumentando a 187 al cabo de 12 días. Para

los días 2, 4, 6, 8, 10 y 12 se presentaron 109, 103, 102, 117, 125 y 116 valores atípicos, respectivamente.

El valor atípico más alto, de 3.73 unidades, se registró el día 12, correspondiente a la serie sin tendencia $\varepsilon^{\theta}_{842}$, con un aumento del error de 3.53 a 13.20 km. Desde el día 10 hasta el día 20, se registraron 166 casos en los que las predicciones del modelo empeoraron los resultados de SGP4. Los valores atípicos fueron 177, 171, 173, 166, 153 y 153. El valor más alto, de 5.02 unidades, se registró el día 20, correspondiente a la serie sin tendencia $\varepsilon^{\theta}_{1412}$, con un incremento del error de 3.01 a 15.11 km.

En el último periodo, solo se registró un valor atípico el día 12, pero a partir de este día los casos aumentaron hasta llegar a 108 el día 22. El valor atípico más alto, de 2.22 unidades, corresponde nuevamente a la serie $\varepsilon^{\theta}_{1412}$. Para este periodo, el error de la predicción pasó de 3.01 a 6.68 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días						
	Días	2	4	6	8	10	12
Mod_1	0	4	23	85	101	132	187
	8	4	42	72	104	126	166
	10	1	15	37	59	85	108

Tabla 4-28.: Casos en los que las predicciones del modelos Mod_1 empeoraron las de SGP4 desde el instante inicial, así como desde los días 8 y 10.

La figura **4-28** muestra los diagramas de caja y bigotes con el análisis de $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para el modelo Mod_1. El horizonte de propagación es de 12 días, comenzando los días 2, 8 y 10. En los tres casos, la diferencia entre el bigote superior e inferior es de 0.33, 0.27 y 0.28 unidades, respectivamente.

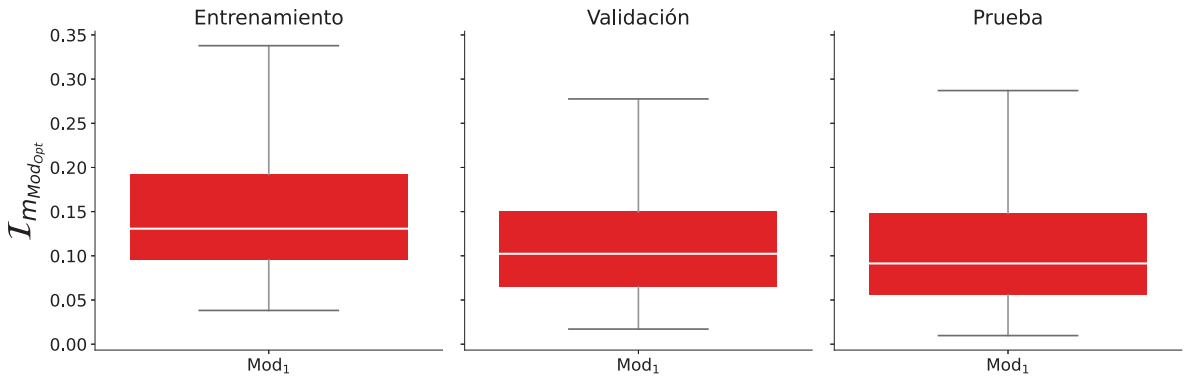


Figura 4-28.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto al óptimo $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para Mod_1 aleatorio en las 1685 series. El horizonte de propagación es de 12 días.

Para las propagaciones desde el día 2, se presentaron 29 valores atípicos. En las propagaciones desde el día 10, se registraron 73 valores atípicos, y en el último caso, 85. Es

importante destacar que, tal como se mencionó en el apartado anterior, aunque el modelo presenta buenos resultados, aún queda un amplio margen para mejoras.

4.3. Optimizando la arquitectura

En la sección anterior se diseñó una arquitectura con dos capas ocultas que contienen 256 y 128 neuronas, respectivamente, mientras que la capa de entrada tiene 168 neuronas. Este modelo de red neuronal se entrenó para mejorar la precisión del propagador orbital SGP4 en TLE de objetos ubicados en la región MEO. Para predecir un nuevo dato, se realiza una multiplicación de una matriz de tamaño 1×168 por una matriz de tamaño 168×256, que contiene los pesos entre las conexiones de la capa de entrada y la primera capa oculta. Después de esta multiplicación, se suman los sesgos y se pasa el resultado como argumento a la función de activación correspondiente. Estas operaciones generan un vector de tamaño 1×256, que se multiplica por la matriz de pesos de tamaño 256×128 que conecta la primera capa oculta con la segunda. Luego, se suman los sesgos de cada neurona y se pasan los resultados a la segunda función de activación. Después de esta operación, se obtiene un vector de tamaño 1×128, que se multiplica por la matriz de pesos que conecta la segunda capa oculta con la capa de salida, que solo tiene una neurona. Finalmente, después de las multiplicaciones, las adiciones y el paso por la última función de activación, se obtiene el valor que indica la predicción de la red neuronal. El tiempo promedio que lleva al modelo de red neuronal de Beronia predecir la serie temporal a 12 días es de aproximadamente 1.73 segundos. Por otro lado, la versión SGP4 de Matlab tarda alrededor de 10 segundos en propagar el TLE en un procesador Intel Xeon E5 a 3.7 GHz.

En esta sección, se busca mejorar la eficiencia computacional del modelo de RN mediante la optimización de su arquitectura. En primer lugar, se reducirá el número de neuronas de la capa de entrada, disminuyendo el número de puntos por revolución de 84 (uno cada 10 minutos) a 12 (uno cada 70 minutos) y, posteriormente, a 6 (uno cada 140 minutos). Una vez establecido el número de puntos por revolución en 6, se reducirá el número de neuronas de las capas ocultas de 256 y 128 a 64 y 32, respectivamente. Es importante recordar que el uso de 12 puntos por revolución, aproximadamente uno cada 70 minutos, fue propuesto inicialmente en [170, 155] para órbitas de tipo LEO y MEO.

4.3.1. Reducir el número de neuronas en la capa de entrada

En esta sección se entrenarán dos grupos de cinco modelos de RN. En el primer grupo, se reducirá el número de puntos por revolución de 84 a 12, lo que equivale a tomar un punto cada 70 minutos (min). En el segundo grupo, el número de puntos se reducirá a 6, es decir, un punto cada 140 min. En adelante, para simplificar la escritura de este documento, se referirá a este valor como la resolución de muestreo o simplemente *resolución*. En el caso donde las muestras son tomadas con una resolución de 70 min, se obtienen 3,803,839 vectores para entrenar y 813,174 para validar el entrenamiento. Con una resolución de 140 min, el número de vectores para entrenar y validar es de 3,807,645 y 816,930, respectivamente. Es importante destacar que el número de vectores de entrenamiento se ha reducido en un 0.19 %

para una resolución de 70 min, mientras que en un 0.09 % para 140 min.

En la figura **4-29** se muestran las gráficas de secuencias de la serie $\varepsilon^{\theta}_{119}$ durante los primeros 4 días de propagación con distintas resoluciones. Esta serie es una de las empleadas para entrenar los modelos. Los datos tomados con una resolución de 10 min se muestran en azul; mientras que en verde y rojo se presentan los datos tomados con una resolución de 70 y 140 min, respectivamente.

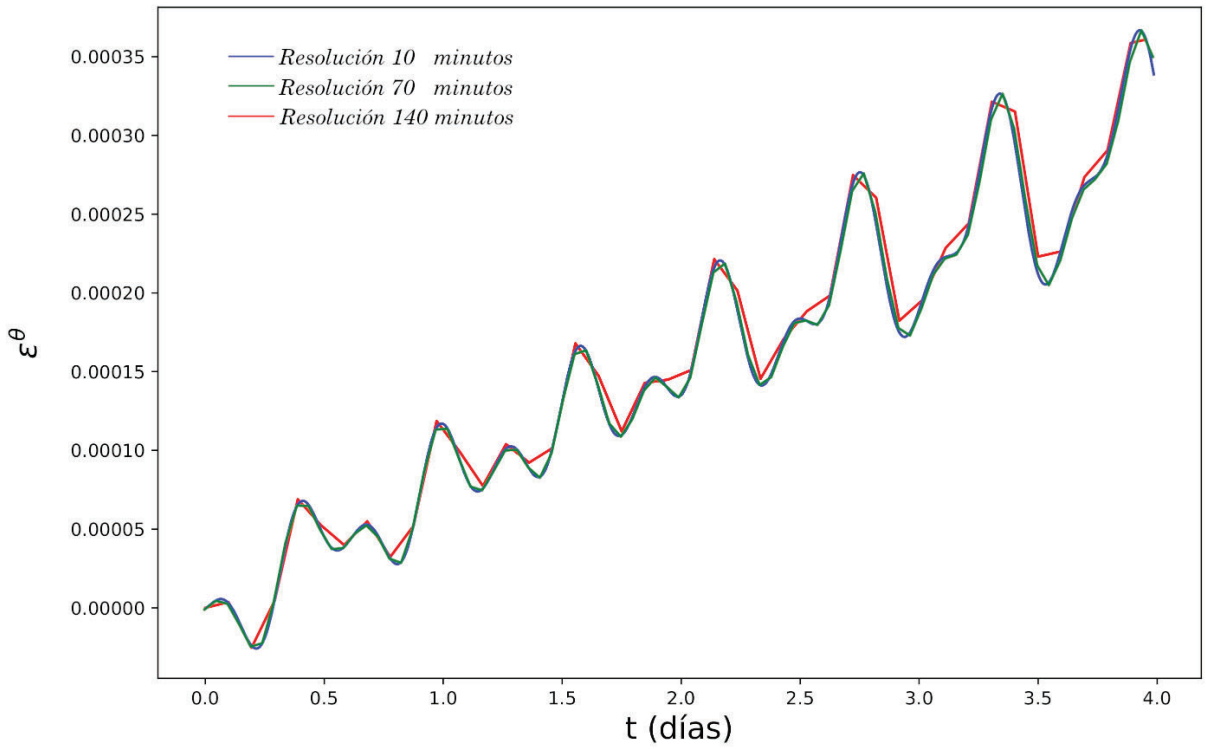


Figura 4-29.: Gráficas de secuencias de la serie $\varepsilon^{\theta}_{119}$. En azul se representa la serie con una resolución de 10 min, en verde con una resolución de 70 min y en rojo con una resolución de 140 min.

En las series con resoluciones de 70 y 140 min se observa cómo se pierden algunos detalles en la definición de la serie a medida que aumenta el tiempo de resolución. Por ejemplo, se pierden las irregularidades entre las oscilaciones y la forma precisa de las ondas. Sin embargo, aún se conservan las características generales de la serie temporal, como la amplitud, el número de oscilaciones y la tendencia.

A continuación se compararán los resultados del entrenamiento de los dos grupos de RN con los resultados obtenidos en el caso anterior. El objetivo es determinar si la pérdida de información al disminuir la resolución de muestreo de las series afecta el entrenamiento de las RN y, por consiguiente, a la precisión de los modelos Mod_i.

Para entrenar las modelos se emplean las 219 series del caso anterior. La única diferencia radica en que en la capa de entrada se reduce el número de neuronas de 168 a 24 para el primer grupo de modelos y a 12 en el segundo. El proceso de evaluación de los modelos

Mod_i se lleva a cabo de manera análoga a como se describió anteriormente. En la tabla **4-29** se muestra el tiempo de cómputo empleado para entrenar cada uno de los cinco modelos. En la primera columna se indica el modelo, en la segunda columna el tiempo que tardó en ser entrenado empleando una resolución de 70 min, y en la tercera columna el tiempo empleado con una resolución de 140 min. Es importante recordar que el corto tiempo de entrenamiento que muestran algunos modelos, por ejemplo, Mod_4, se debe a la activación del hiperparámetro *paciencia*, que detuvo su entrenamiento mucho antes de alcanzar el número máximo de épocas permitido. Al igual que en la primera iteración, el entrenamiento y las predicciones de cada uno de estos modelos se ejecutaron en el clúster de supercomputación Beronia, utilizando 28 CPU.

	Resolución	
Modelo	70 min	140 min
1	17.20	10.73
2	14.61	8.48
3	15.12	9.85
4	9.73	8.73
5	8.92	16.40

Tabla 4-29.: Tiempo de cómputo en días de los modelos de RN entrenados con una resolución de 70 y 140 min.

En la figura **4-30** se representan los diagramas de caja y bigotes que analizan los máximo errores en distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos de RN entrenados con una resolución de 70 min, para un horizonte de propagación de 12 días. En la tabla **4-30** se muestra el número de valores atípicos, que no se incluyen en los diagramas para facilitar la visualización de los resultados.

Modelo	Número de valores atípicos - Tiempo en días					
	2	4	6	8	10	12
Mod_1	0	0	0	1	1	3
Mod_2	0	6	5	0	0	1
Mod_3	0	0	0	0	0	0
Mod_4	0	2	1	0	0	0
Mod_5	0	0	0	0	0	0

Tabla 4-30.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-30** según el día de propagación.

La diferencia entre el bigote superior e inferior para Mod_5, que presentó los valores más altos, fue de 2.34, 6.06, 11.46, 19.24, 27.82 y 48.69 km para los 2, 4, 6, 8, 10 y 12 días, respectivamente. Los valores máximos alcanzados por este modelo fueron de 2.96, 7.16, 12.84, 21.58, 30.64 y 52.32 km, reduciendo el error de SGP4 en aproximadamente 13.38, 21.87, 26.97, 25.16, 27.61 y 14.01 km, respectivamente.

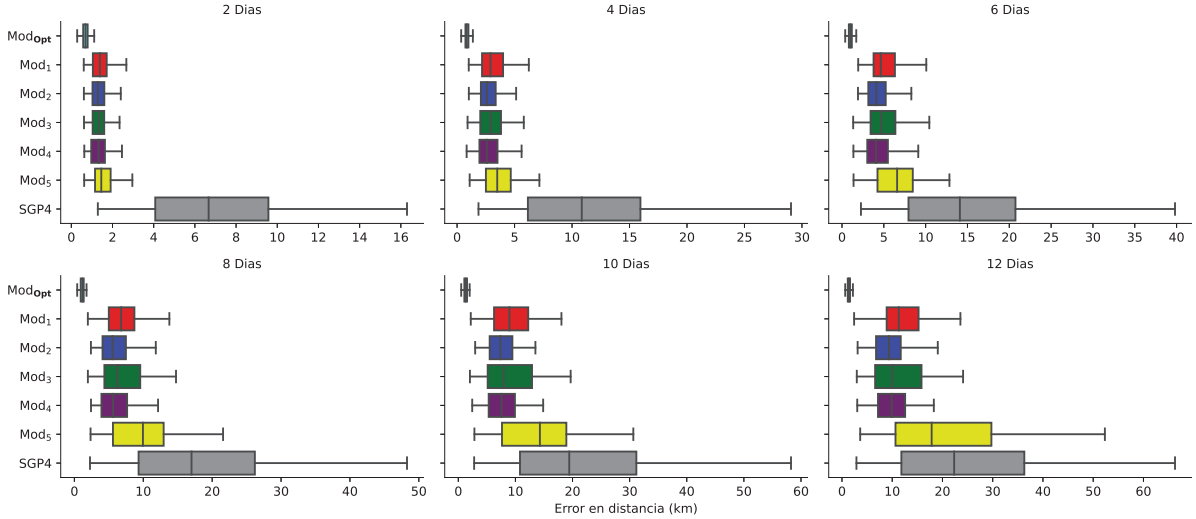


Figura 4-30.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN entrenados con una resolución de 70 min, con los datos de la fase de entrenamiento.

Por otro lado, los errores más pequeños en el día 2 de propagación son registrados por Mod_3 con 2.34 km. Del día 4 al día 10, Mod_2 con 5.14, 8.29, 11.82 y 13.5 km, respectivamente, y en el día 12, Mod_4 con 18.28 km. Comparado con los errores más pequeños cometidos por los modelos aleatorios con una resolución de 10 min, se observa un incremento en el error en los días 2, 4 y 8 de 0.48, 0.55 y 0.69 km, respectivamente, y una reducción de 0.02, 1.62 y 0.08 km en los días 6, 10 y 12.

Durante el tiempo de propagación, los Mod_3 y Mod_5 no presentaron valores atípicos. Mod_1 presentó 5 valores, de los cuales 3 se observaron el día 12. El modelo Mod_2, con 12 valores, presentó el número más alto, de estos, 6 se detectaron el día 4 y 5 el día 6. Por su parte, Mod_4 solo presentó 3 valores atípicos.

Para concluir, el tiempo de ejecución promedio de la predicción del modelo se reduce a aproximadamente 0.84 segundos, lo que representa poco menos de la mitad del tiempo requerido por los modelos con una resolución de 10 min.

En la tabla **4-31** se muestra el número de casos en los que las predicciones de los modelos empeoraron las de SGP4. El modelo Mod_5, con 95 casos (un 43.16 % de sus predicciones), mostró el peor comportamiento después de 12 días de propagación. Sin embargo, los modelos Mod_2 y Mod_4, con 35 casos, presentaron el mejor rendimiento. En comparación con los 22 casos presentados por el modelo Mod_1 aleatorio con una resolución de 10 min, el rendimiento de los modelos basados en RN se redujo en 13 casos. El índice de mejora más alto lo presentó Mod_3 (que obtuvo el error más pequeño) al día 12, con 3.7 unidades. Cabe destacar que este índice pertenece a la serie sin tendencia $\varepsilon^{\theta}_{140}$, en la que el error a 12 días se incrementó de 2.89 a 10.69 km.

En la figura **4-31** se presentan los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos de RN entrenados

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	2	4	6	8	10	12
Mod_1	3	20	39	44	51	58
Mod_2	1	9	22	27	32	35
Mod_3	2	11	21	35	47	56
Mod_4	1	8	19	29	31	35
Mod_5	4	22	38	57	80	95

Tabla 4-31.: Numero de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 70 min, empeoraron las de SGP4 durante el periodo de entrenamiento.

con una resolución de 140 min, para un horizonte de propagación de 12 días. La tabla **4-32** muestra el número de valores atípicos. El tiempo que tarda cada uno de los modelos en realizar una predicción completa es de 0.27 segundos, aproximadamente un tercio del tiempo que tardan los modelos con una resolución de 70 min.

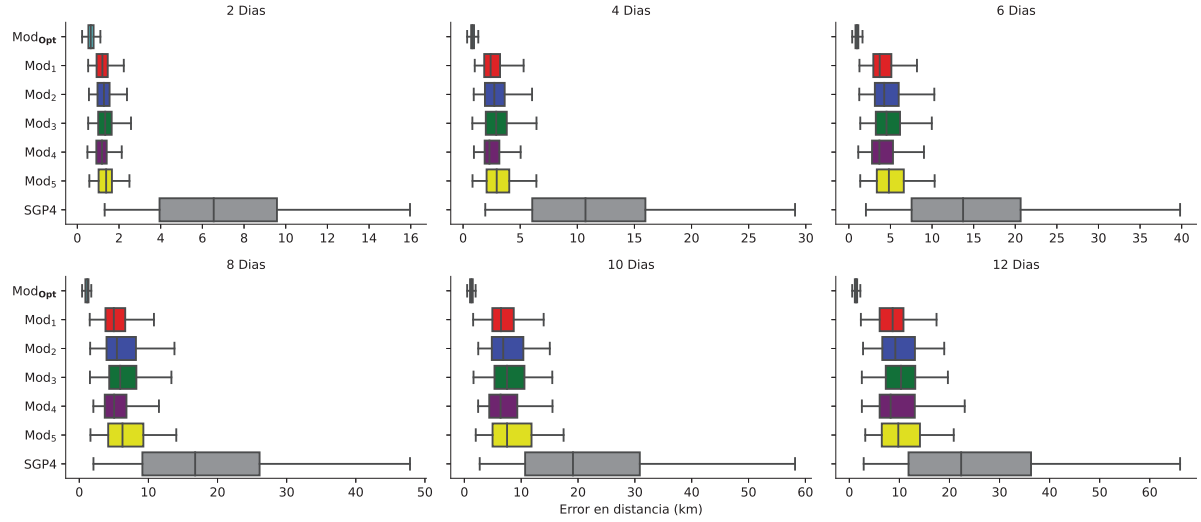


Figura 4-31.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN entrenados con una resolución de 140 min, con los datos de la fase de entrenamiento.

La diferencia entre los bigotes superior e inferior en el caso máximo de los modelos Mod_i es de 2.06, 5.61, 8.94, 12.41, 15.45 y 17.67 km durante los 12 días de propagación. Los errores más altos se alcanzan en Mod_5, desde el día 2 al día 10, con 2.58, 6.43, 10.31, 14.03 y 17.46 km. En el día 12, el error más grande se observa en Mod_4, con 23.05 km. En comparación con los modelos entrenados con una resolución de 70 min, se puede ver una reducción del máximo error en 0.38, 0.73, 2.53, 7.55, 13.18 y 29.27 km del día 2 al 12, respectivamente. Los errores más pequeños se encuentran en Mod_4 con 2.13 y 5.04 km durante los primeros 4 días de propagación. Durante los días posteriores, el error más pequeño se alcanzan en Mod_1 con 8.19, 10.8, 13.96 y 17.41 km. Respecto a los errores más pequeños presentes en los modelos

Modelo	Número de valores atípicos - Tiempo en días					
	2	4	6	8	10	12
Mod_1	22	19	15	15	6	2
Mod_2	8	12	17	20	25	25
Mod_3	12	13	14	19	17	13
Mod_4	6	11	16	18	11	11
Mod_5	5	12	21	31	37	11

Tabla 4-32.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-31** según el día de propagación.

entrenados con una resolución de 70 min, se observa una reducción de 0.21, 0.1, 0.1, 1.02 y 0.87 km para los días 2, 4, 6, 8 y 12, respectivamente. En el día 10, se incrementa el error en 0.46 km. En comparación con el modelo Mod_1 aleatorio entrenado con una resolución de 10 min, se incrementa el error en 0.27 y 0.45 km en los primeros 4 días, pero se reduce en 0.12, 0.33, 1.16 y 0.95 km en los días restantes. En este caso, se han incrementado notablemente los valores atípicos. El modelo Mod_5 con 37 datos presenta el mayor número entre los modelos al día 10 de propagación, mientras que al día 12, es Mod_2 con 25 datos.

En la tabla **4-33** se enumeran los casos donde las predicciones de los modelos entrenados con una resolución de 140 min empeoraron los resultados de SGP4. Según los resultados mostrados en la tabla, los modelos Mod_1 y Mod_4 muestran el mejor comportamiento. Por ejemplo, al día 12, el número de casos en que empeoran los resultados de SGP4 son 29 y 32 respectivamente. Por otro lado, Mod_3, presenta el mayor número de casos donde empeoraron los resultados de SGP4, con 54 casos al día 12.

Comparado con lo resultados de los modelos entrenados con una resolución de 70 min, donde a los 12 días Mod_5 presentó 95 casos, se ha mejorado el rendimiento de los modelos en 43 casos. Con esta resolución, el índice de mejora más alto observado en el modelo Mod_1 (que obtuvo el error más pequeño) al día 12 fue de 2.02 unidades. Este valor corresponde, al igual que en el caso anterior, a la serie sin tendencia $\varepsilon^{\theta}_{140}$, cuyo error se incrementó de 2.89 a 5.83 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	2	4	6	8	10	12
Mod_1	1	9	15	21	28	29
Mod_2	2	8	29	33	42	46
Mod_3	3	13	34	40	49	54
Mod_4	1	8	16	21	27	32
Mod_5	3	15	35	44	52	52

Tabla 4-33.: Número de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 140 min, empeoraron las de SGP4 durante el periodo de entrenamiento.

En la figura **4-32** se muestran los diagramas de caja y bigotes de los errores en distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos Mod_i entrenados con una resolución

de 70 min a partir del décimo día de propagación. Además, el número de valores atípicos de cada modelo se muestra en la tabla **4-34**.

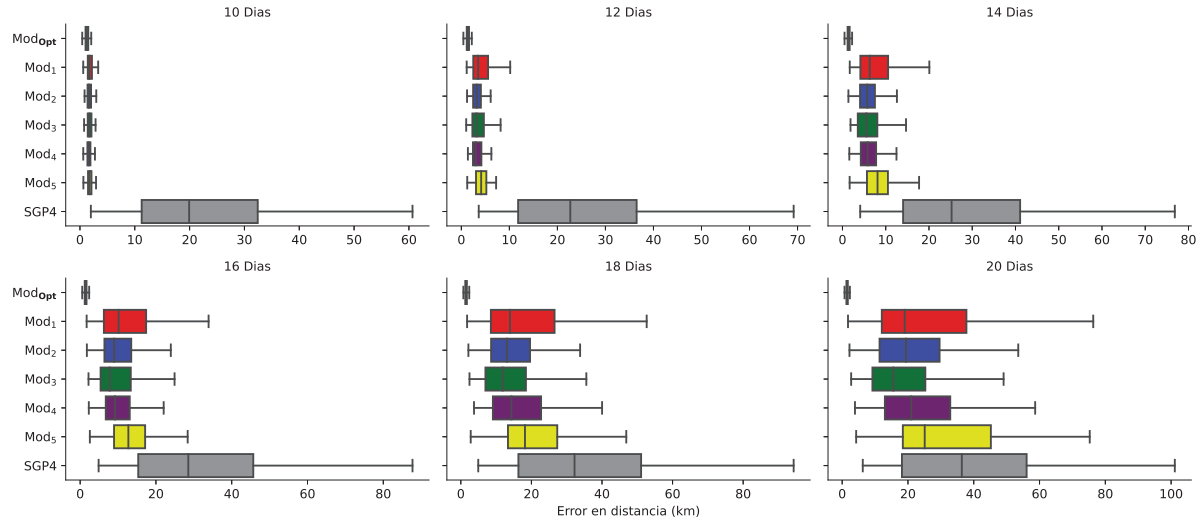


Figura 4-32.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{Opt}$ y los cinco modelos de RN entrenados con una resolución de 70 min. Las propagaciones comienzan en la fase de validación.

Modelo	Número de valores atípicos - Tiempo en días					
	10	12	14	16	18	20
Mod$_1$	22	19	15	15	6	2
Mod$_2$	8	12	17	20	25	25
Mod$_3$	12	13	14	19	17	13
Mod$_4$	6	11	16	18	11	11
Mod$_5$	5	12	21	31	37	11

Tabla 4-34.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-32** según el día de propagación.

La mayor diferencia entre el bigote superior e inferior se observa en Mod$_1$, con 2.72, 9.06, 18.4, 32.18, 50.89 y 74.55 km para los días 10 a 20, respectivamente. En comparación con SGP4, se aprecia una reducción en la dispersión de al menos 53.54, 56.47, 54.5, 50.78, 38.6 y 19.52 km durante el periodo de propagación. Este modelo también registró los errores más altos, con 3.32, 10.17, 20.08, 33.9, 52.66 y 76.33 km. Sin embargo, el error más pequeño se alcanza en Mod$_4$ con 2.84 km en el día 10, en el día 12 en Mod$_2$ con 6.10 km, en los días 14 y 16 en Mod$_4$ con 12.49 km y 22.04 km, respectivamente, en el día 18 en Mod$_2$ con 33.8 km, y finalmente en el día 20 en Mod$_3$ con 49.07 km.

En comparación con los mínimos errores del modelo Mod$_1$ aleatorio entrenado con una resolución de 10 min, se observa un incremento en estos errores de 0.19, 0.70, 1.79, 4.48, 9.15 y 17.82 km. La mayor concentración de valores atípicos la presenta Mod$_5$ con 37 valores en

el día 18. En el día 20 de propagación, Mod_2 con 25 valores, presentó el número más alto. Por su parte, Mod_1 presentó el número más bajo, con solo 2 valores en el día 20.

En la tabla **4-35** se muestra el número de casos donde las predicciones de los modelos con una resolución de 70 min empeoraron a SGP4. Durante los primeros días de propagación, el modelo Mod_4 presenta el peor comportamiento con 2 y 10 casos en los días 10 y 12, respectivamente. A partir de este día, Mod_5 es el que presenta el peor comportamiento, empeorando los resultados de SGP4 en hasta 65 casos el día 20.

En comparación con el modelo Mod_2 aleatorio con una resolución de 10 min en el día 20, se observa un aumento de 25 casos en los cuales se empeoran los pronósticos de SGP4. El índice de mejora más alto mostrado por Mod_3, que obtuvo los errores más pequeños, en el día 20 fue de 3.37 unidades y pertenece a la serie de tendencia positiva $\varepsilon^{\theta}_{310}$. En esta serie, el error en distancia se incrementó de 8.70 a 29.32 km, mientras que la tendencia de la serie es de 2.96×10^{-07} unidades (ver tabla **4-3**). Es decir, aunque se clasificó como tendencia positiva, su pendiente está muy cerca de la de las series clasificadas como series sin tendencia.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	10	12	14	16	18	20
Mod_1	1	3	10	18	22	31
Mod_2	1	1	8	18	34	45
Mod_3	1	2	11	19	23	30
Mod_4	2	10	16	24	32	43
Mod_5	1	9	20	35	45	65

Tabla 4-35.: Número de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 70 min, empeoraron las de SGP4 durante el periodo de validación.

La figura **4-33** presenta los diagramas de caja y bigotes que ilustran los errores en distancia entre AIDA y SGP4, Mod_{Opt}, y cada uno de los modelos Mod_i entrenados con una resolución de 140 min a partir del décimo día de propagación. En la tabla **4-36** se muestra el número de valores atípicos. Antes de iniciar el análisis, es fundamental señalar que los resultados del Mod_4 no se incluyen en la figura. Esto se debe a que dicho modelo presento un sobreajuste a los datos de entrenamiento, lo que generó errores en la distancia muy altos en las predicciones sobre los datos de validación. Como resultado, la visualización de los resultados de los modelos restantes se ve afectada.

Modelo	Número de valores atípicos - Tiempo en días					
	10	12	14	16	18	20
Mod_1	6	11	14	17	21	23
Mod_2	11	18	35	36	39	48
Mod_3	5	14	20	26	35	39
Mod_5	6	14	10	16	20	25

Tabla 4-36.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-33** según el día de propagación.

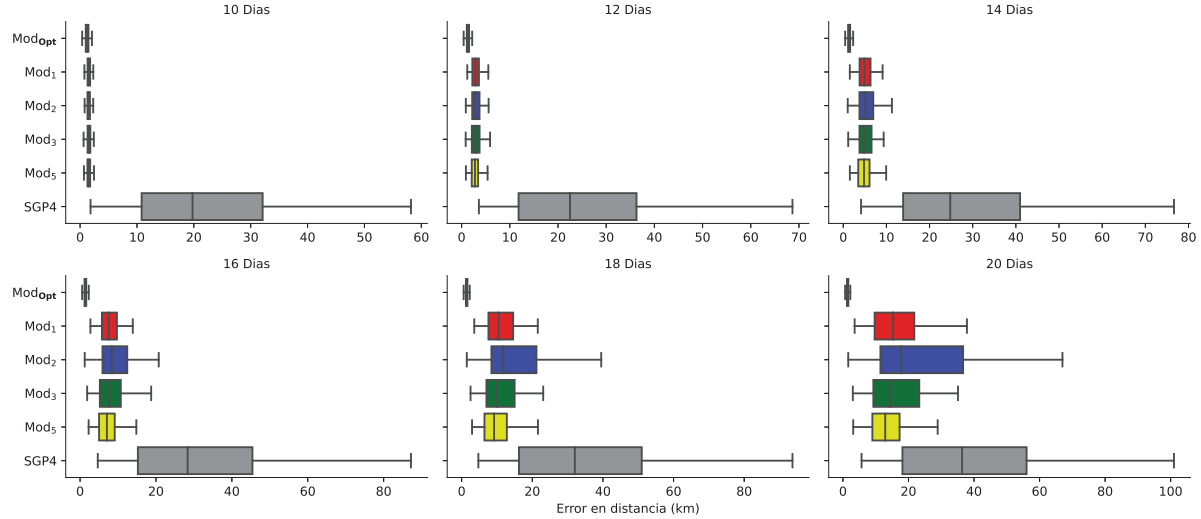


Figura 4-33.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y los cinco modelos de RN entrenados con una resolución de 140 min. Las propagaciones comienzan en la fase de validación.

El modelo Mod$_2$ presenta la mayor diferencia entre los bigotes superior e inferior, con 1.76, 5.06, 10.24, 19.5, 38.06 y 65.36 km para los días 10, 12, 14, 16, 18 y 20, respectivamente. Los errores más altos para los días 10 y 12 se observan en Mod$_3$, con 2.45 y 5.91 km. Para los días restantes, los errores más elevados los presenta Mod$_2$, con 11.28, 20.73, 39.52 y 66.98 km. Al comparar los errores máximos con los modelos entrenados con una resolución de 70 min, se observa una reducción del error de 0.87, 4.26, 8.8, 13.17, 13.14 y 9.35 km del día 10 al 20. Los errores más bajos se registran en Mod$_1$ desde el día 10 hasta el 18, con 2.31, 5.38, 9.11, 13.89 y 21.56 km. Para el día 20, el error más bajo se encuentra en Mod$_5$, con 28.91 km. En comparación con los errores más bajos de los modelos entrenados con una resolución de 70 min, se logra una reducción de 0.53, 0.72, 3.38, 8.15, 12.24 y 20.16 km. Respecto al modelo Mod$_1$ aleatorio entrenado con una resolución de 10 min, la reducción es de 0.34, 0.02, 1.59, 3.67, 3.09 y 2.34 km. Sin embargo, Mod$_2$ presenta el mayor número de valores atípicos durante los 12 días de propagación, alcanzando los 48 valores en el día 20.

En la tabla **4-37** se detallan los casos en que las predicciones de los modelos empeoraron los resultados de SGP4. Comparando con los modelos entrenados con una resolución de 70 min, se observa un mejor comportamiento de estos modelos durante los primeros 4 días. Sin embargo, en el día 12, el modelo Mod$_3$ muestra un peor rendimiento, aunque empeora en la mitad de los casos que el Mod$_4$ entrenado con una resolución de 70 min. Para el día 20 de propagación, Mod$_3$ presenta 68 casos en los que los resultados empeoran en comparación con SGP4, lo que representa un aumento de 3 casos en relación con Mod$_5$, que exhibió el peor desempeño entre los modelos entrenados con una resolución de 70 min. El índice de mejora más alto del Mod$_5$ (que tuvo los errores más pequeños) en el día 20 fue de 3.0 unidades y corresponde a la serie de tendencia positiva $\varepsilon^{\theta}_{226}$, donde el error de predicción aumentó de 6.39 a 19.16 km. Esta serie muestra una pendiente poco pronunciada de $1.27{\times}10^{-07}$, lo que

la acerca a las series clasificadas como sin tendencia (ver tabla **4-3**).

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	10	12	14	16	18	20
Mod_1	0	1	13	25	31	50
Mod_2	0	1	15	43	59	75
Mod_3	1	5	21	37	53	68
Mod_5	1	2	10	20	33	46

Tabla 4-37.: Número de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 140 min, empeoraron las de SGP4 durante el periodo de validación.

La figura **4-34** muestra los diagramas de caja y bigotes que representan los errores en distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos Mod_i entrenados con una resolución de 70 min a partir del duodécimo día de propagación. En la tabla **4-38** se presenta el número de valores atípicos.

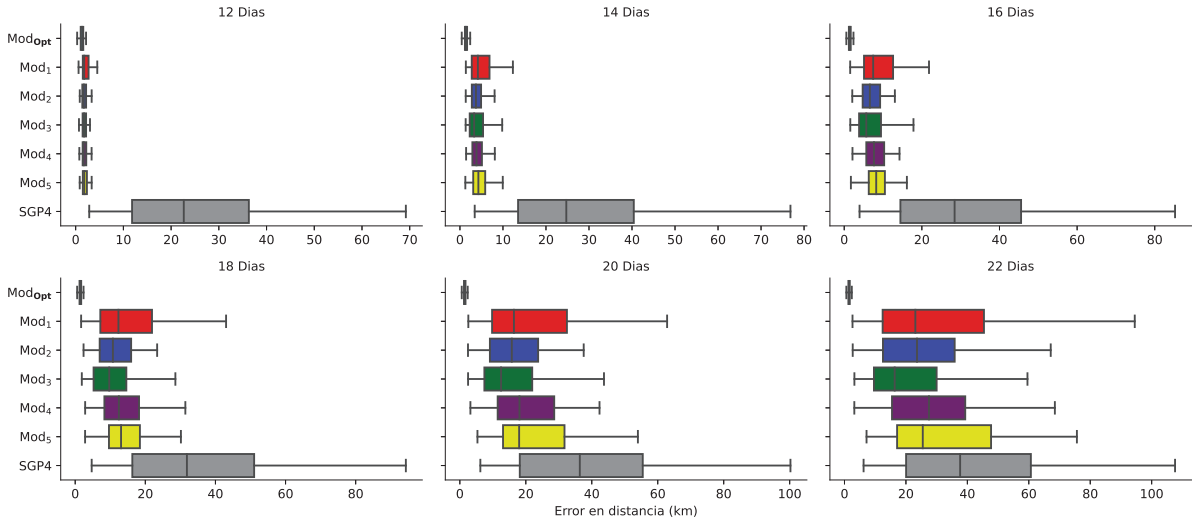


Figura 4-34.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN entrenados con una resolución de 70 min. Las propagaciones comienzan en la fase de prueba.

En primer lugar, se destaca que Mod_1 exhibe la mayor diferencia entre el bigote superior e inferior, con valores de 3.92, 10.94, 20.3, 41.35, 60.26 y 91.88 km para los días del 12 al 22, respectivamente. Asimismo, este modelo registra los errores más altos, con valores de 4.53, 12.31, 21.84, 43.04, 62.84 y 94.49 km para los mismos días. Esto supone una reducción significativa de los errores más altos en comparación con SGP4 por parte de todos los modelos, de al menos 61.8, 64.74, 63.42, 51.34, 37.51 y 12.84 km.

El modelo Mod_3 alcanza el error más pequeño de 3.00 km en el día 12. A partir del día 14 hasta el día 20, los errores más pequeños son registrados por Mod_2, con 8.06, 13.05, 23.37 y

Modelo	Número de valores atípicos - Tiempo en días					
	12	14	16	18	20	22
Mod_1	22	16	25	14	10	4
Mod_2	18	21	29	31	31	30
Mod_3	25	24	25	26	20	16
Mod_4	16	20	25	24	11	11
Mod_5	16	20	30	36	35	11

Tabla 4-38.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-34** según el día de propagación.

37.55 km. Nuevamente, en el día 22, el error más pequeño se registra en Mod_3, con 59.56 km. Comparado con los errores más pequeños cometidos por el modelo aleatorio Mod_1 entrenado con una resolución de 10 min, se observa una reducción de 0.29 km para el día 12 y de 2.53 km para el día 16. Sin embargo, para los días 14, 18, 20 y 22, el error se incrementa en 0.42, 0.11, 8.48 y 19.67 km, respectivamente.

En cuanto a los valores atípicos, durante los primeros 4 días de propagación, Mod_3 muestra el mayor número, con 25 y 24 valores. Entre los días 16 y 20, el mayor número de valores atípicos lo presenta Mod_5, con 30, 36 y 35 valores. Finalmente, en el día 22, el mayor número de valores atípicos se observa en Mod_2.

La tabla **4-39** muestra el número de casos en los que las predicciones de los modelos empeoraron los resultados de SGP4. Los tres primeros modelos de la tabla muestran un buen comportamiento durante los primeros 4 días de propagación. Por ejemplo, en los primeros 2 días, ninguno de los modelos superó el error en distancia de SGP4, y durante los siguientes 2 días, el máximo número de casos registrados fue de 6. Sin embargo, a partir del día 16, el número de casos se incrementó notablemente en todos los modelos, alcanzando Mod_5 un máximo de 73 casos.

El índice de mejora más alto, mostrado por Mod_3 en el día 22, fue de 3.82 unidades y pertenece nuevamente a la serie de tendencia positiva $\varepsilon^{\theta}_{226}$. En esta serie, el error en distancia se incrementó de 6.39 a 24.40 km en 22 días. Por el contrario, este modelo presentó los errores más pequeños.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	12	14	16	18	20	22
Mod_1	0	6	15	24	30	36
Mod_2	0	5	8	27	45	51
Mod_3	0	6	12	24	30	40
Mod_4	9	12	18	23	29	41
Mod_5	8	14	26	33	43	73

Tabla 4-39.: Número de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 70 min, empeoraron las de SGP4 durante el periodo de validación.

La figura **4-35** muestra los diagramas de caja y bigotes que representan los errores en

distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos Mod_i entrenados con una resolución de 140 min a partir del duodécimo día de propagación. El número de valores atípicos se presenta en la tabla **4-40**.

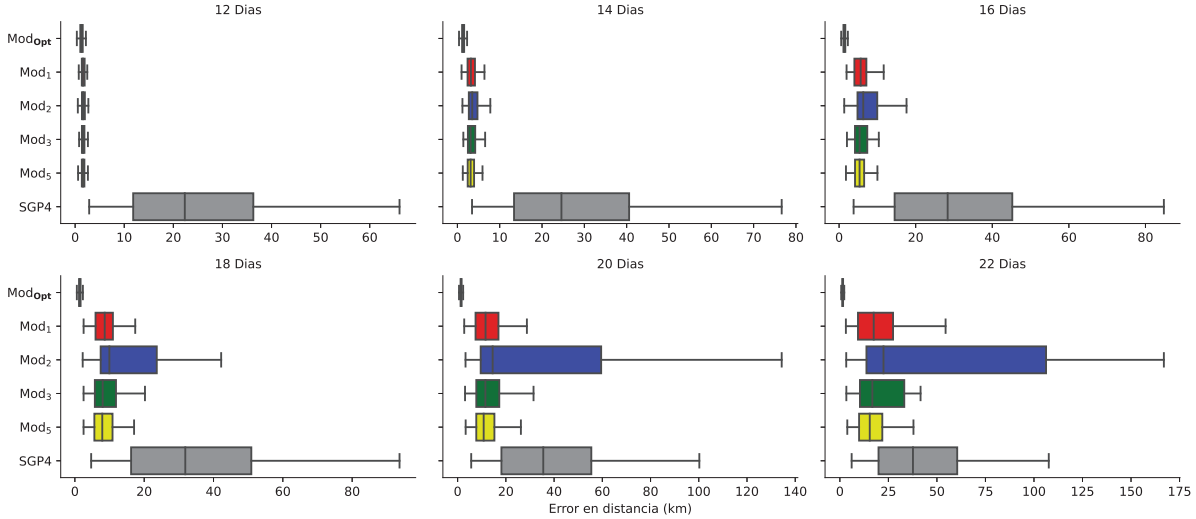


Figura 4-35.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN entrenados con una resolución de 140 min. Las propagaciones comienzan en la fase de prueba.

Modelo	Número de valores atípicos - Tiempo en días					
	12	14	16	18	20	22
Mod_1	13	15	15	19	19	23
Mod_2	26	38	42	42	26	0
Mod_3	19	23	29	38	39	48
Mod_5	11	12	15	20	23	27

Tabla 4-40.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-35** según el día de propagación.

La mayor dispersión se observa en el modelo Mod_2, con una diferencia entre el bigote superior e inferior de 2.14, 6.58, 16.26, 39.95, 131.11 y 163.64 km para los días 12, 14, 16, 18, 20 y 22, respectivamente. En comparación con SGP4, la dispersión se reduce del día 12 al 18 en 61.28, 66.94, 65.05 y 49.73 km, pero se incrementa durante los días 20 y 22 en 37.02 y 62.57 km, respectivamente. Este modelo también muestra los errores más altos, con valores de 2.69, 7.78, 17.61, 42.21, 134.39 y 166.92 km. Por otro lado, los errores más pequeños se observan en Mod_5, con valores de 2.47, 5.93, 10.01, 17.08, 26.24 y 37.91 km para los días 12, 14, 16, 18, 20 y 22, respectivamente. Comparado con los mínimos errores de los modelos entrenados con una resolución de 70 min, se observa una reducción de 0.53, 2.13, 3.04, 6.29, 11.31 y 21.65 km. Frente al Mod_1 aleatorio entrenado con una resolución de 10 min, el mínimo error se reduce en 0.82, 1.71, 5.57, 6.18, 2.83 y 1.98 km desde los días 12 al 22.

Durante los primeros 4 días de propagación, el Mod_5 muestra el menor número de valores atípicos. En el día 16, comparte junto con Mod_1 el menor número de valores atípicos. Desde el día 18 hasta el 22, es Mod_1 el que presenta la menor cantidad de valores atípicos. Por otro lado, en el día 22, el modelo con más valores atípicos es Mod_3. Sin embargo, Mod_2, a pesar de haber tenido los errores de propagación más altos y la mayor dispersión, no muestra valores atípicos para el día 22.

En la tabla **4-41** se muestra el número de casos en los que las predicciones de los modelos entrenados con una resolución de 140 min empeoraron los resultados de SGP4. Durante los primeros 2 días de propagación (día 12), ninguno de los modelos empeoró las predicciones de SGP4. En los días siguientes, destacaron los modelos Mod_1 y Mod_5 por su buen comportamiento. Por ejemplo, en el día 14, estos modelos solo superaron el error de SGP4 en 2 y 4 casos, respectivamente. En el día 22, Mod_5 mostró el mejor rendimiento, con solo 46 casos en los que se observó un empeoramiento. Sin embargo, se evidencia un mal comportamiento por parte de Mod_2, ya que en este caso se observaron 97 casos (un 44 % de sus predicciones) en los que se empeoraron los resultados de SGP4. Comparando los resultados del día 22 con el modelo Mod_1 entrenado con una resolución de 70 min, se observa un aumento en el número de casos en los que se empeoran las predicciones de SGP4 en 10 casos. En comparación con el modelo Mod_1 aleatorio y una resolución de 10 min, el número de casos se incrementa en 23. El índice de mejora más alto mostrado por Mod_5, que presenta los errores más pequeños, en el día 22 fue de 3.43 unidades y pertenece nuevamente a la serie de tendencia positiva $\varepsilon^{\theta}_{226}$, donde el error de predicción se incrementó de 6.39 a 15.59 km en 22 días.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	12	14	16	18	20	22
Mod_1	0	2	19	21	35	53
Mod_2	0	13	35	57	77	97
Mod_3	0	15	33	51	57	74
Mod_5	0	4	17	29	37	46

Tabla 4-41.: Número de casos en los que las predicciones de los modelos Mod_i, entrenados con una resolución de 140 min, empeoraron las de SGP4 durante el periodo de validación.

Comparación entre los modelos entrenados con una resolución de 70 y 140 minutos

Al reducir a la mitad el tamaño del vector de entrada en las RN con una resolución de 140 min, el tiempo necesario para el pronóstico se reduce aproximadamente a un tercio del tiempo requerido con una resolución de 70 minutos.

En la tabla **4-42**, se comparan los errores mínimos obtenidos para cada uno de los dos grupos de cinco modelos. Como se puede observar, en los modelos entrenados con una resolución de 140 min, los errores son menores en las tres fases evaluadas. No obstante, en favor de los modelos entrenados con una resolución de 70 min, se registra un menor número de casos donde las predicciones de SGP4 empeoran durante las fases de validación y prueba (30 casos para los modelos de 70 minutos y 46 para los de 140 minutos). A pesar de esto, los

modelos entrenados con una resolución de 140 min logran índices de mejora más bajos. Por ejemplo, en el caso más desfavorable, el modelo que ofreció los mejores resultados entrenado con esta resolución, incrementó el error de SGP4 de 6.39 a 19.16 km en la fase de validación. Por el contrario, en el mejor de los modelos entrenados con una resolución de 70 min, el error se incrementó desde 8.70 hasta 29.32 km.

En resumen, con los modelos entrenados con una resolución de 140 min, se requieren menos operaciones para realizar las predicciones, lo que se traduce en propagaciones más rápidas (0.25 segundos en Beronia), errores mínimos más pequeños y menores incrementos en el error de distancia en comparación con los modelos entrenados con una resolución de 70 min. Por lo tanto, se selecciona esta resolución para la siguiente y última fase de optimización, que consiste en reducir la arquitectura de la red neuronal a 64 y 32 neuronas en las dos capas ocultas.

Resolución	70 minutos						140 minutos					
	2d	4d	6d	8d	10d	12d	2d	4d	6d	8d	10d	12d
Entrenamiento	2.34	5.14	8.29	11.82	13.50	18.28	2.13	5.04	8.19	10.8	13.96	17.41
Validación	2.84	6.10	12.49	22.04	33.80	49.07	2.31	5.38	9.11	13.89	21.56	28.91
Prueba	3.00	8.06	13.05	23.37	37.55	59.56	2.47	5.93	10.01	17.08	26.24	37.91

Tabla 4-42.: Mínimos errores en distancia (en km) obtenidos por los modelos entrenados con una resolución de 70 min en comparación con los modelos entrenados con una resolución de 140 min.

4.3.2. Reducir el número de neuronas de las capas ocultas

En este último apartado, se procedió a reentrenar cinco modelos de redes neuronales con una arquitectura que consta de una capa de entrada de 12 neuronas, dos capas ocultas con 64 y 32 neuronas respectivamente, y una capa de salida con una sola neurona. Se utilizaron 3,807,645 vectores para la fase de entrenamiento y 816,930 para la fase de validación. La tabla **4-43** muestra el tiempo de cómputo empleado en el entrenamiento de cada modelo. Es importante destacar que en el caso de Mod_4, se volvió a activar el hiperparámetro *paciencia*, dado que no mostró mejoría en el aprendizaje antes de alcanzar el número máximo de épocas permitido.

Modelo	Tiempo (días)
1	8.2
2	8.9
3	8.2
4	5.9
5	6.8

Tabla 4-43.: Tiempo de cómputo de los cinco modelos de RN.

En la figura **4-36** se presentan los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y SGP4, Mod_{Opt} y cada uno de los modelos de redes

neuronales entrenados para un horizonte de propagación de 12 días. La tabla **4-44** muestra el número de valores atípicos observados. Además, se destaca que el tiempo necesario para realizar una predicción con estos modelos es de 0.19 segundos, lo que representa una mejora de aproximadamente 0.10 segundos en comparación con los modelos entrenados con una resolución de 140 min en el caso anterior.

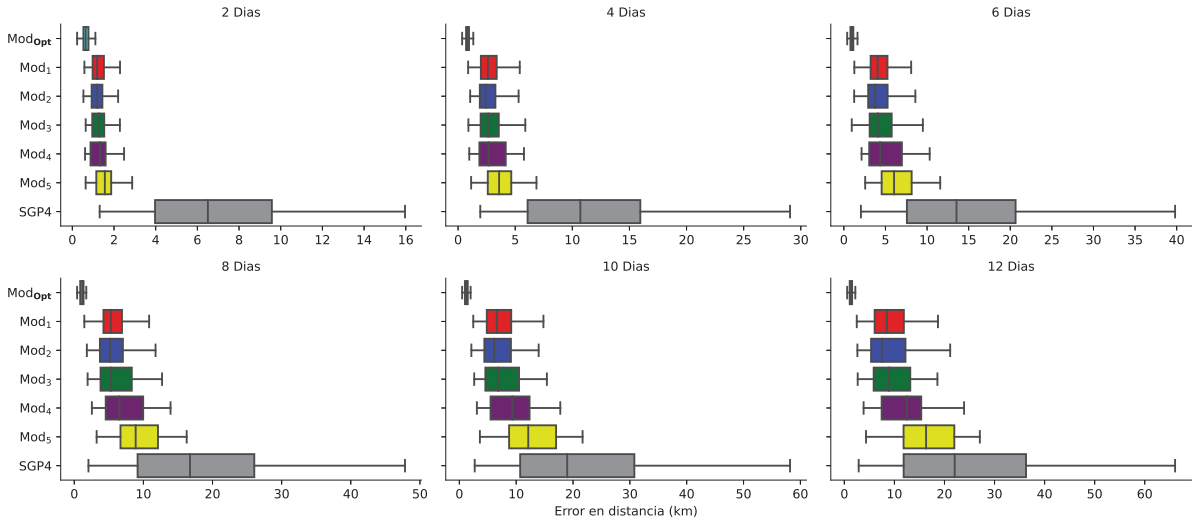


Figura 4-36.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\mathrm{Opt}}$ y los cinco modelos de RN entrenados con los datos de la fase de entrenamiento.

Modelo	Número de valores atípicos - Tiempo en días					
	2	4	6	8	10	12
Mod$_1$	10	4	1	1	0	0
Mod$_2$	11	4	3	3	3	3
Mod$_3$	11	5	5	5	6	9
Mod$_4$	10	4	1	1	0	0
Mod$_5$	9	6	7	15	17	21

Tabla 4-44.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-36** según el día de propagación.

La máxima diferencia entre los bigotes superior e inferior de los modelos Mod$_i$ es de 2.23, 5.72, 9.01, 13.02, 18.06 y 22.73 km durante los 12 días de propagación. Lo que supone una reducción en la dispersión respecto a SGP4 de 12.8, 21.36, 28.52, 31.44, 37.43 y 40.71 km, respectivamente. Los errores más altos para estos días se observaron en el modelo Mod$_5$, alcanzando los 2.87, 6.86, 11.58, 16.26, 21.68 y 27.06 km, respectivamente. En comparación con los modelos entrenados con una resolución de 140 min, el máximo error se incrementó en 0.29, 0.43, 1.27, 2.23, 4.22 y 4.01 km.

Los errores más pequeños se registraron en Mod$_2$ en los días 2, 4 y 10, con valores de 2.2, 5.3 y 13.95 km, respectivamente. En los días 6 y 8, Mod$_1$ mostró los errores más bajos,

con 8.09 y 10.83 km, respectivamente. Finalmente, en el día 12, el error más pequeño se observó en Mod_3, con 18.59 km. En comparación con los errores más pequeños observados en los modelos entrenados con una resolución de 140 min en el caso anterior, se observa un incremento de 0.07, 0.26, 0.03 y 1.18 km en los días 2, 4, 8 y 12, respectivamente, y una reducción de 0.1 y 0.01 km en los días 6 y 10.

Durante los 2 primeros días de propagación, los modelos presentaron entre 9 y 11 valores atípicos. A partir del cuarto día, Mod_5 registró el mayor número de valores atípicos, alcanzando 21 para el día 12. Por el contrario, los modelos Mod_1 y Mod_4 no mostraron valores atípicos en el último día de propagación.

La tabla **4-45** muestra el número de casos en los cuales las predicciones de los modelos empeoraron los errores de SGP4. Durante los primeros 2 días de propagación, Mod_2 no presentó ninguna predicción que empeorara los resultados de SGP4. Por otro lado, Mod_5 mostró los peores resultados, con un incremento en el error de SGP4 de hasta 71 casos en el día 12. A partir del cuarto día de propagación, los modelos Mod_1 y Mod_4 presentaron el menor número de casos, con tan solo 6. Mientras tanto, Mod_2 proporcionó los mejores resultados a partir del sexto día de propagación, con 13, 19, 25 y 29 casos, respectivamente. En comparación con los resultados de los modelos entrenados con una resolución de 140 min, donde el mayor número de casos se observó en Mod_3 con 54 casos, se obtiene una reducción de 17 casos. El índice de mejora más alto observado en Mod_3 (que obtuvo el error más pequeño) en el día 12 fue de 4.08 unidades, correspondientes a la serie $\varepsilon^{\theta}_{140}$, donde el error se incrementó de 2.89 a 11.79 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	2	4	6	8	10	12
Mod_1	1	6	22	25	25	32
Mod_2	0	7	13	19	25	29
Mod_3	2	7	23	32	33	31
Mod_4	1	6	22	25	25	32
Mod_5	3	19	43	54	65	71

Tabla 4-45.: Número de casos en los que las predicciones de los modelos Mod_i empeoraron las de SGP4 durante el periodo de entrenamiento.

El diagrama de caja y bigotes de los errores en la distancia entre AIDA y SGP4, Mod_{Opt}, y cada uno de los modelos Mod_i a partir del décimo día de propagación se muestra en la figura **4-37**. En la tabla **4-46** se presenta el número de valores atípicos de cada modelo.

La mayor diferencia entre el bigote superior e inferior se observa en Mod_5, desde el día 10 hasta el día 18, con 2.11, 6.24, 13.54, 23.73, y 36.85 km, respectivamente. Para el día 20, la mayor diferencia se observa en Mod_1 con 54.32 km. En comparación con las máximas diferencias de SGP4, se observa una reducción en la dispersión de aproximadamente 54.15, 59.29, 59.37, 59.12, 52.6, y 39.75 km. Mod_5 también alcanza los errores más altos de propagación del día 10 al 18, con 2.69, 7.54, 15.25, 25.61 y 39.0 km. En el día 20, el error más alto se observa en Mod_1 con 56.74 km. Sin embargo, en los modelos entrenados con una resolución de 140 min, el error se incrementa en 0.24, 1.63, 3.97, y 4.88 km del día 10 al día 16, pero se reduce en 0.52 y 10.24 km para los días restantes.

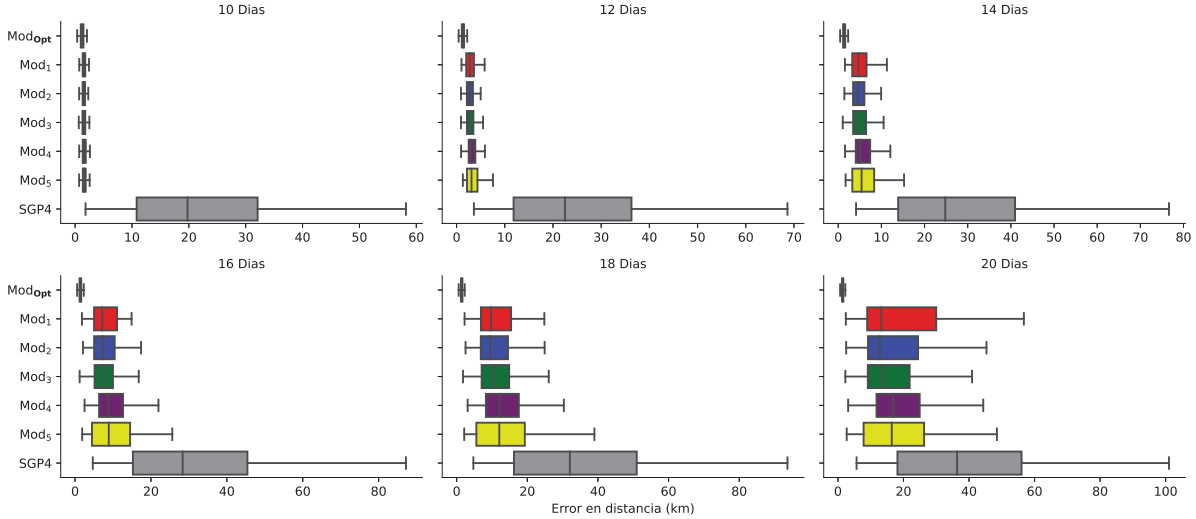


Figura 4-37.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y los cinco modelos de RN entrenados con los datos de la fase de validación.

Modelo	Número de valores atípicos - Tiempo en días					
	10	12	14	16	18	20
Mod$_1$	18	11	4	8	23	34
Mod$_2$	19	15	14	6	5	2
Mod$_3$	18	12	15	16	14	11
Mod$_4$	18	14	12	11	13	17
Mod$_5$	15	18	15	16	15	16

Tabla 4-46.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-37** según el día de propagación.

Por otro lado, los errores más pequeños se observan en Mod$_2$ durante los días 10 a 14, con 2.21, 5.36 y 9.94 km, respectivamente. Para los días 16 y 18, en el Mod$_1$, se registran 14.9 y 22.9 km, respectivamente. Finalmente, en el día 20 de propagación, el error más pequeño se observa en Mod$_3$, con un máximo de 39.35 km. En comparación con los mínimos errores cometidos por los modelos entrenados con una resolución de 140 min, se observa una reducción de 0.10 y 0.02 km en los primeros 4 días de propagación (días 10 y 12), mientras que para los días restantes, el error se incrementa en 0.83, 1.01, 1.34 y 10.44 km."

Durante los primeros dos días de propagación, los modelos registran entre 15 y 19 valores atípicos, siendo Mod$_2$ el que presenta el número más alto. Sin embargo, para los días 18 y 20, Mod$_1$ alcanzó el mayor número de valores atípicos, con 23 y 34, respectivamente. Por último, cabe destacar que Mod$_2$ presenta el menor número de valores atípicos en los días 18 y 20, con solo 5 y 2, respectivamente

En la tabla **4-47** se muestran los casos en los que las predicciones de los modelos empeoraron los resultados de SGP4. Durante los dos primeros días de propagación (día 10), solo

se registró un caso en el que Mod_5 empeoró las predicciones de SGP4. Sin embargo, para los días 12 y 14, Mod_5 presenta el mayor número de casos, con 3 y 12, respectivamente. A partir del día 16, Mod_1 muestra un comportamiento menos favorable, con 30, 52 y 70 casos, respectivamente. Por otro lado, en el día 20, Mod_3 destaca por su buen rendimiento, ya que solo 17 de sus predicciones empeoraron los resultados de SGP4. El índice de mejora más alto observado en Mod_3 para el día 20 es de 2.45 unidades, correspondiente a la serie $\varepsilon^{\theta}_{226}$, donde el error se incrementó de 6.39 a 15.66 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	10	12	14	16	18	20
Mod_1	0	1	6	30	52	70
Mod_2	0	1	9	26	39	51
Mod_3	0	1	6	15	17	17
Mod_4	0	1	8	19	32	64
Mod_5	1	3	12	18	23	36

Tabla 4-47.: Número de casos en los que las predicciones de los modelos Mod_i empeoraron las de SGP4 durante el periodo de validación.

En la figura **4-38** se muestran los diagramas de caja y bigotes de los errores en la distancia entre AIDA y SGP4, así como entre AIDA y Mod_{Opt}, junto con cada uno de los modelos Mod_i entrenados a partir del duodécimo día de propagación. El número de valores atípicos se presenta en la tabla **4-48**.

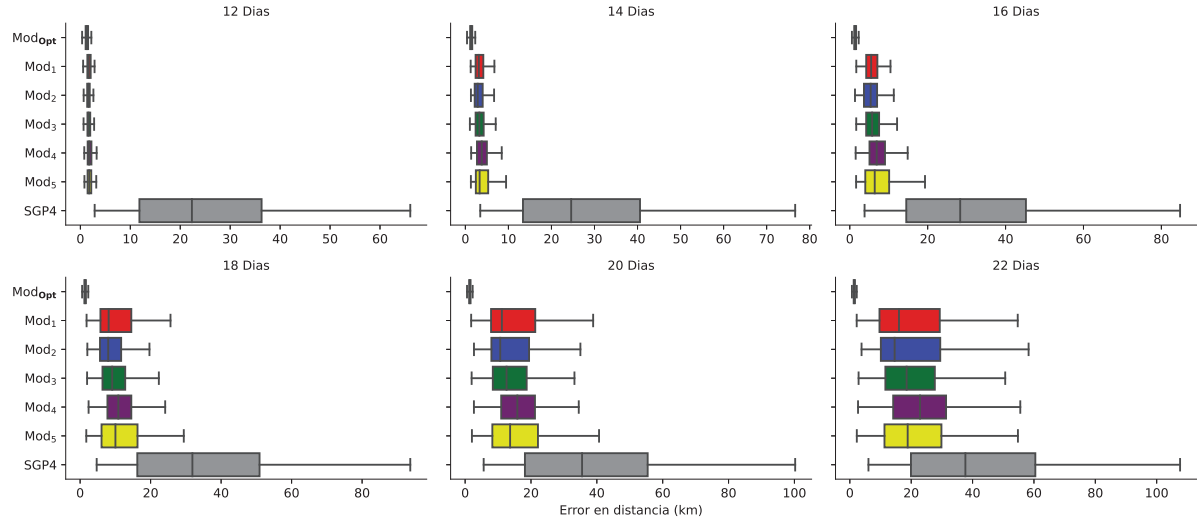


Figura 4-38.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y los cinco modelos de RN entrenados con los datos de la fase de prueba.

Se observa que en los días 12, 14, 16, 18 y 20, Mod_5 exhibe una dispersión notable, con una diferencia entre el bigote superior e inferior de 2.37, 8.17, 16.46, 27.66 y 38.61 km,

Modelo	Número de valores atípicos - Tiempo en días					
	12	14	16	18	20	22
Mod_1	17	16	9	19	22	31
Mod_2	13	13	16	5	2	1
Mod_3	14	15	15	14	9	8
Mod_4	13	9	10	10	12	20
Mod_5	19	15	14	15	16	15

Tabla 4-48.: Número de valores atípicos de los diagramas de caja y bigotes de la figura **4-37** según el día de propagación.

respectivamente. Sin embargo, al llegar al día 22, es Mod_2 quien muestra la mayor dispersión, alcanzando 54.44 km. Al comparar esta variabilidad con la de SGP4 durante esta fase de propagación, se aprecia una reducción significativa de 61.05, 65.36, 64.85, 62.02, 55.48 y 46.63 km, respectivamente.

Los errores más pequeños se registran en Mod_2 durante los días 12, 14 y 18, con valores de 2.61, 5.93 y 19.67 km, respectivamente. Para el día 16, Mod_1 muestra un error mínimo de 10.45 km, mientras que en los días 20 y 22, Mod_3 exhibe errores mínimos de 32.73 y 50.64 km, respectivamente. Comparando estos mínimos con los errores de los modelos entrenados con una resolución de 140 min en el caso anterior, se observa un incremento de 0.14, 0.44, 2.59, 6.49 y 12.73 km los días 12, 16, 18, 20 y 22, respectivamente, manteniéndose constante el error el día 14.

Durante los primeros 2 días de propagación, los modelos Mod_2 y Mod_3 presentaron el menor número de valores atípicos (13), mientras que Mod_5 registró el máximo número de valores atípicos (19). Al llegar al día 22 de propagación, Mod_2 mostró solo un valor atípico, mientras que Mod_4 presentó el mayor número, con 20 valores atípicos.

En la tabla **4-49** se muestra el número de casos en los que las predicciones de los modelos empeoraron los resultados de SGP4. Durante los primeros dos días de propagación, se registró un caso para los modelos Mod_1 y Mod_2, mientras que los modelos restantes no presentaron ninguno. Para el día 14, Mod_5 muestra el número más alto de casos, con 7, seguido de Mod_3 con 4 casos. Al llegar al día 16, Mod_2 presenta 17 casos, siendo este el valor más alto para ese día. En los siguientes días, se observa un aumento en el número de casos en todos los modelos, donde las predicciones empeoran en comparación con SGP4. Para el día 22, el mayor número de casos se registra en Mod_4, con 63 casos. Por otro lado, Mod_3 muestra el menor número de casos, con 14. El índice de mejora más alto registrado por Mod_3 (que mostró el error más pequeño) al día 22, es de 2.63 unidades y pertenece a la serie de tendencia positiva $\varepsilon^{\theta}_{226}$, donde el error aumentó de 6.39 a 16.81 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días					
	12	14	16	18	20	22
Mod_1	1	4	15	33	44	57
Mod_2	0	2	17	30	40	49
Mod_3	0	4	5	11	13	14
Mod_4	0	3	9	18	43	63
Mod_5	1	7	14	17	24	45

Tabla 4-49.: Número de casos en los que las predicciones de los modelos Mod_i empeoraron las de SGP4 durante el periodo de prueba.

Selección del mejor modelo

Al igual que en el caso de los modelos secuenciales, la elección del mejor modelo está determinado por el período de propagación seleccionado. Para predicciones que requieran una alta precisión a corto plazo, Mod_2 resulta ser una excelente opción. Este modelo exhibió los errores más bajos durante los primeros cuatro días en las fases de entrenamiento, validación y prueba, así como la menor diferencia entre los valores de los bigotes superior e inferior. A mediano plazo, Mod_1 se postula como el modelo más adecuado debido a su comportamiento. Obtuvo los errores más pequeños, así como la menor dispersión, en los días 6 y 8 en la fase de entrenamiento, los días 16 y 18 en la fase de validación, y el día 16 en la fase de prueba. Para predicciones a largo plazo, Mod_3 se destaca por su rendimiento. Durante la fase de entrenamiento, logró el error más pequeño al día 12. En la fase de validación, destacó en el día 20, y en la fase de prueba, mostró su mejor comportamiento en los días 20 y 22.

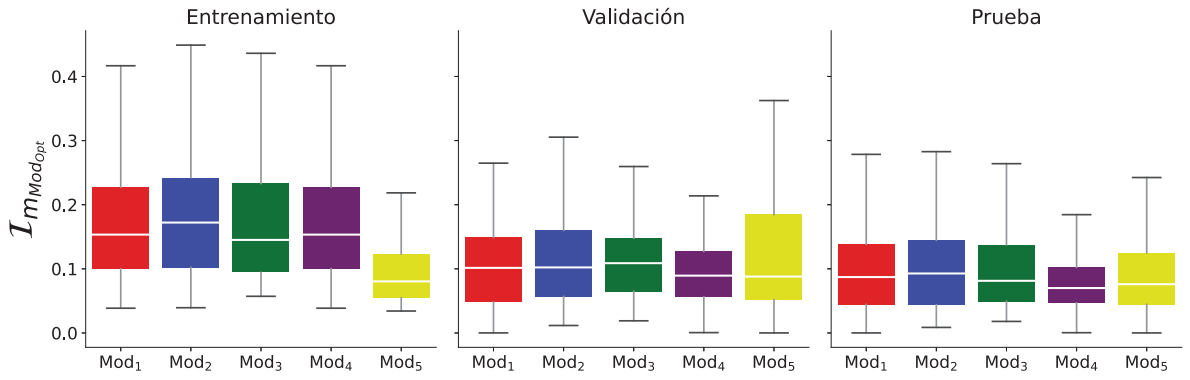


Figura 4-39.: Diagramas de caja y bigotes con el análisis del índice de mejora respecto al óptimo $\mathcal{I}m_{\text{Mod}_{\text{Opt}}}$ para los modelos Mod_i, en un horizonte de propagación de 12 días con los datos de entrenamiento, validación y prueba.

En la figura **4-39**, se presentan los diagramas de caja y bigotes que analizan el $\mathcal{I}_{m_{\text{Mod}_{\text{Opt}}}}$ para los modelos de RN con un horizonte de propagación de 12 días en los conjuntos de entrenamiento, validación y prueba (mostrados de izquierda a derecha, respectivamente). Al observar la figura, se puede verificar que, en los datos de entrenamiento, los cuatro primeros modelos muestran un comportamiento similar. Esto se evidencia en los bigotes superiores,

que se sitúan entre 0.4 y 0.45 unidades, y los valores del cuartil Q_3, que oscilan entre 2.2 y 2.5 unidades. Sin embargo, en los conjuntos de validación y prueba, destaca el hecho de que Mod_2 muestra el bigote inferior y el cuartil Q_1 más altos entre los modelos entrenados.

Robustez del modelo entrenado

Para evaluar la robustez de Mod_3, se emplearán las 93 series que no fueron utilizadas durante el proceso de entrenamiento del modelo. Este análisis tiene como propósito examinar la capacidad del modelo para identificar patrones similares a los procesados durante la etapa de entrenamiento, así como para realizar predicciones basadas en estos patrones, junto con otros desconocidos. Para ello, se generarán predicciones de las series y se analizará su precisión desde el inicio de la propagación, así como en los días 8 y 10, que corresponden al inicio de los conjuntos de validación y prueba utilizados para entrenar el modelo. Para evaluar el comportamiento del modelo Mod_3 en este análisis, se volverá a calcular el error en la distancia entre AIDA y HEncke$_{\mathbf{SGP4}}$. El horizonte de propagación considerado será de 12 días, y los errores se registrarán en los días 2, 4, 6, 8, 10 y 12 posteriores al inicio de las predicciones.

Análisis a partir del día 2

En la figura **4-40**, se muestran los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod_{Opt} y Mod_3 para un horizonte de propagación de 12 días.

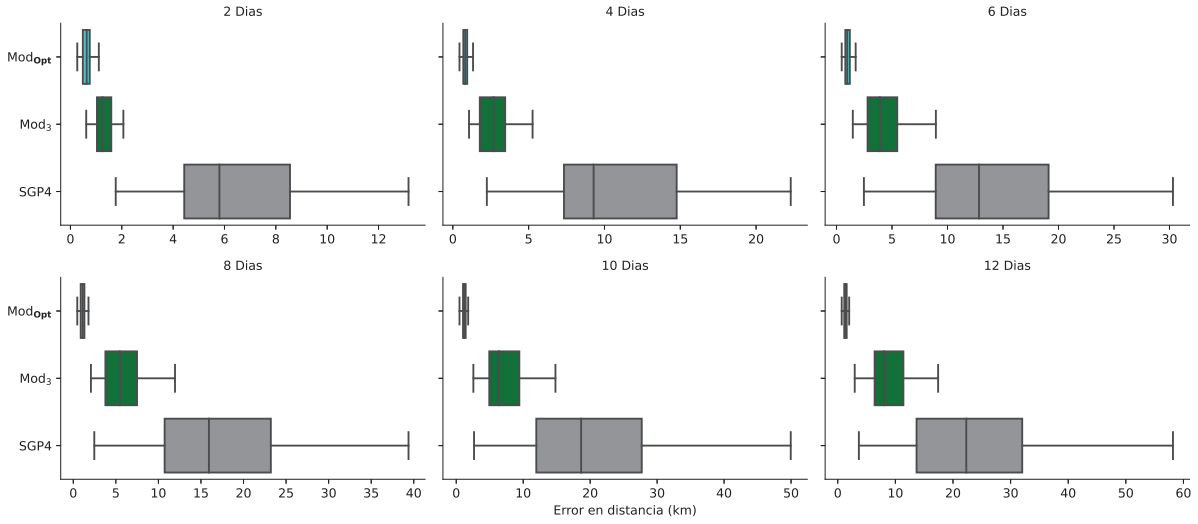


Figura 4-40.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y el Mod_3 de las 93 series. Las predicciones inician a partir del día 2.

La diferencia entre el bigote superior e inferior para SGP4 en los días 2, 4, 6, 8, 10 y 12 es de 11.64, 20.0, 27.67, 36.41, 47.21 y 54.35 km, respectivamente. Con Mod_3, esta diferencia se

reduce a aproximadamente 1.4, 4.20, 7.48, 9.88, 12.26 y 14.48 km. El máximo error alcanzado por SGP4 en los mismos días es de 13.2, 22.33, 30.32, 39.07, 49.96 y 58.27 km, mientras que el máximo error alcanzado por Mod_3 es de 2.06, 5.26, 8.95, 11.95, 14.82 y 17.42 km. Esto representa una reducción de aproximadamente 11.14, 17.07, 21.37, 27.12, 35.14 y 40.85 km. Sin embargo, frente al modelo óptimo, la diferencia en los mismos instantes se incrementa en 0.95, 3.93, 7.23, 10.16, 13.03 y 15.44 km. En comparación con las predicciones del Mod_1 entrenado con una resolución de 10 min, el máximo error se incrementa en 0.27, 0.67, 0.64 y 1.04 km durante los primeros 8 días, pero se reduce en 0.09 y 1.07 km los días 10 y 12, respectivamente. Durante los 12 días de propagación, el máximo error de las predicciones de Mod_3 no supera la mediana de SGP4, que se ubica alrededor de los 23.64 km. En cuanto a los valores atípicos, solo se presentaron 3 valores en el día 2 y 5 en el día 1

Análisis a partir del día 10

La figura **4-41** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod_{Opt} y Mod_3 para un horizonte de propagación de 12 días a partir del décimo día de propagación.

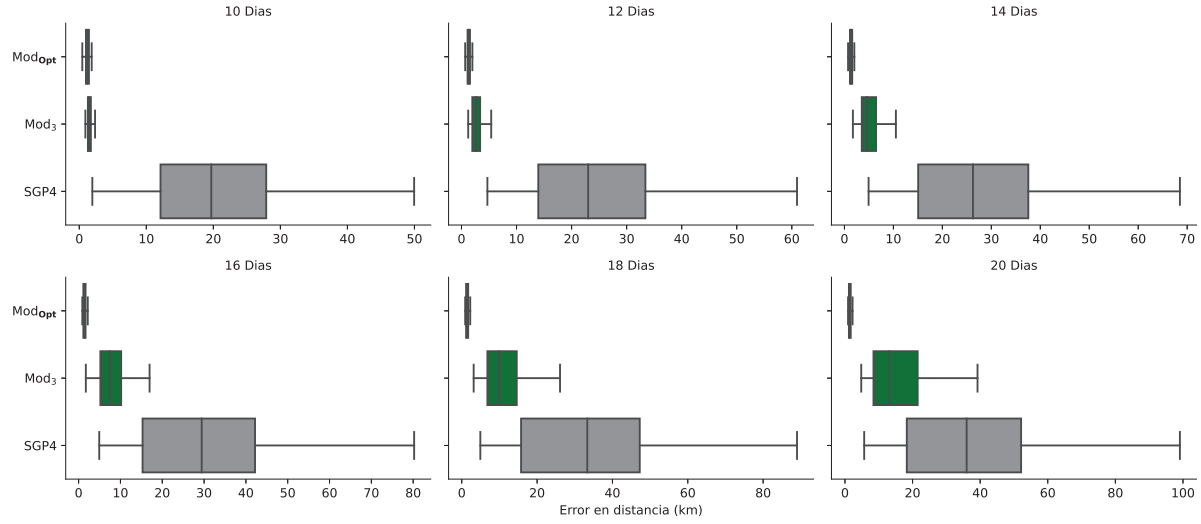


Figura 4-41.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y el Mod_3 en las 93 series. Las predicciones inician a partir del día 10.

Las diferencias entre los bigotes superior e inferior de SGP4 son aproximadamente de 47.38, 56.25, 63.82, 75.42, 84.19 y 91.42 km para los días 10, 12, 14, 16, 18 y 20, respectivamente. Sin embargo, esta dispersión se reduce significativamente en el modelo Mod_3, a alrededor de 1.45, 4.18, 8.78, 15.24, 22.92 y 36.32 km. Los máximos errores alcanzados por el modelo entrenado son aproximadamente de 2.22, 5.40, 10.51, 16.97, 26.09 y 41.14 km. Esto implica una reducción considerable de aproximadamente 47.74, 55.68, 58.41, 63.54, 63.19 y 57.56 km en comparación con SGP4. Frente al modelo óptimo, el error se incrementa en 0.51, 3.42,

8.49, 14.78, 23.84 y 38.89 km, respectivamente. Al comparar con las predicciones del Mod_1 entrenado con una resolución de 10 min, el máximo error se reduce en 0.55 km en el día 10, pero se incrementa en 0.74, 1.6, 1.8, 6.12 y 12.93 km en los días restantes. Durante esta fase, el modelo presentó 2 valores atípicos en el día 16 y 11 en el día 18. Es importante destacar que el valor del máximo error del modelo no supera el valor de Q_3 de SGP4, que se ubica alrededor de los 49.95 km en el día 20.

Análisis a partir del día 12

La figura **4-42** muestra los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod_{Opt} y Mod_3 para un horizonte de propagación de 12 días a partir del duodécimo día de propagación.

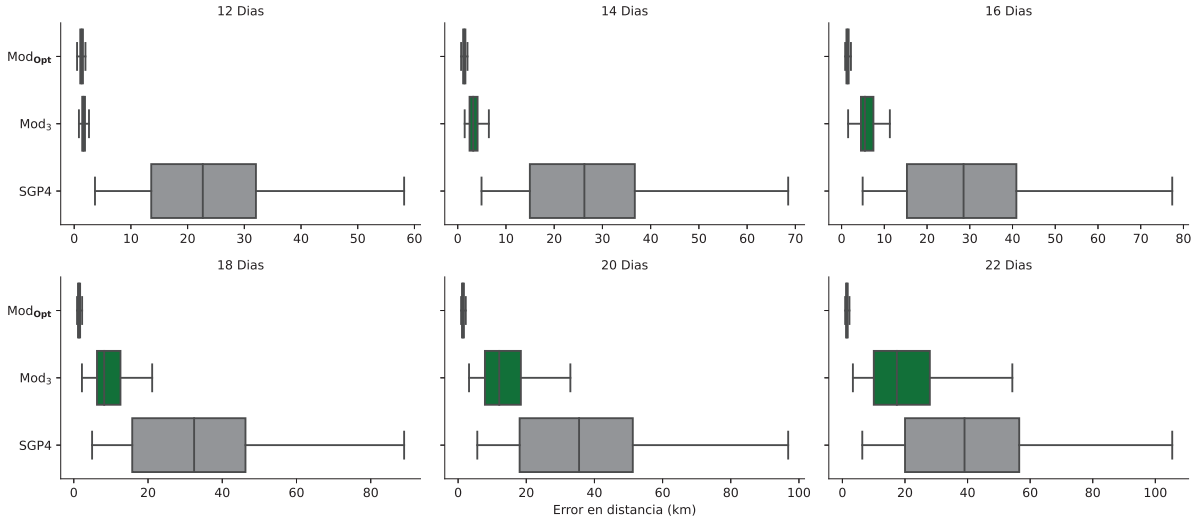


Figura 4-42.: Diagramas de caja y bigotes de los máximos de los errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y el Mod_1 en las 93 series. Las predicciones inician a partir del día 12.

En esta caso, la diferencia entre los bigotes superior e inferior de SGP4 es de aproximadamente 54.74, 63.82, 72.6, 84.19, 91.03 y 98.15 km para los días 12, 14, 16, 18, 10 y 22, respectivamente. Por otro lado, en el caso del modelo Mod_3, esta dispersión se reduce a alrededor de 1.73, 5.01, 10.32, 20.41, 28.97 y 47.77 km. Los máximos errores cometidos por Mod_3 se sitúan aproximadamente en 2.56, 6.46, 11.84, 22.61, 32.17 y 51.13 km. Frente a los máximos errores de SGP4, se observa una considerable reducción de aproximadamente 56.10, 62.46, 65.85, 66.67, 65.13 y 53.64 km, respectivamente. Sin embargo, frente al modelo óptimo, aún se mantiene una distancia de 0.62, 4.44, 9.65, 20.36, 29.92 y 48.88 km, respectivamente. Al comparar con los errores máximos cometidos por Mod_1 entrenado con una resolución de 10 min, observamos una reducción de alrededor de 0.42 km en el día 12, pero un incremento de 0.41, 0.29, 5.36, 9.99 y 22.63 km en los días restantes. Durante esta fase, se presentaron 4 valores atípicos en el día 12, 6 en el día 14, 3 en el día 16 y 8 en el día 22.

Índices de mejora

En la tabla **4-50** se muestra el número de casos en los que las predicciones del Mod_1 empeoraron los resultados de SGP4. Durante los dos primeros días de propagación, no se observó ningún caso en el cual las predicciones del modelo fueran inferiores a las de SGP4. En este período, el índice de mejora más alto alcanzado fue de 2.74 unidades, perteneciente a la serie sin tendencia $\varepsilon^{\theta}_{52}$, donde el error aumentó de 3.91 a 10.71 km. En las propagaciones realizadas a partir del octavo día, no se registraron casos durante los primeros dos días. En los días 12 y 14, se registró un caso por cada día. Para el día 20 se registraron 6 casos, siendo el índice más alto de 2.17 unidades, correspondiente a la serie sin tendencia $\varepsilon^{\theta}_{99}$, donde el error se incrementó de 6.27 a 13.61 km. En el último periodo de propagación, los primeros casos en los que las predicciones empeoraron los resultados de SGP4 aparecieron el día 18, con 2 casos. Para el día 22, se registraron 5 casos, siendo el índice de mejora más alto ese día de 1.72 unidades, observado nuevamente en la serie $\varepsilon^{\theta}_{99}$, donde el error aumentó de 7.62 a 13.10 km.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días						
Mod_3	Día	2	4	6	8	10	12
		0	5	8	7	9	11
	Día 8	10	12	14	16	18	20
		0	1	1	3	3	6
	Día 10	12	14	16	18	20	22
		0	0	0	2	4	5

Tabla 4-50.: Número de casos en los que las predicciones del modelo Mod_3 empeoraron las de SGP4 desde el instante inicial, así como desde los días 8 y 10.

4.3.3. Generalización del modelo entrenado

Finalmente, se evalúa la capacidad de generalización del Mod_3 utilizando las 1685 series generadas a partir de los datos del satélite GALILEO-FM3. Como se mencionó anteriormente, estas series muestran un comportamiento similar a las utilizadas en el entrenamiento del modelo, pero también presentan patrones que no han sido completamente capturados por las redes neuronales. El proceso de evaluación sigue un enfoque análogo al descrito para los modelos aleatorios de la primera iteración. Se mantiene un horizonte de propagación de 12 días, y los errores son registrados en los días 2, 4, 6, 8, 10 y 12 posteriores al inicio de las predicciones.

Análisis a partir del día 2

En la figura **4-43** se muestran los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod_{Opt} y Mod_3 para cada uno de los 1685 TLE.

La diferencia entre los bigotes superior e inferior para SGP4 en los días 2, 4, 6, 8, 10 y 12 es de 18.69, 34.04, 46.84, 62.8, 75.44 y 87.9 km, respectivamente. Por otro lado, en el caso del

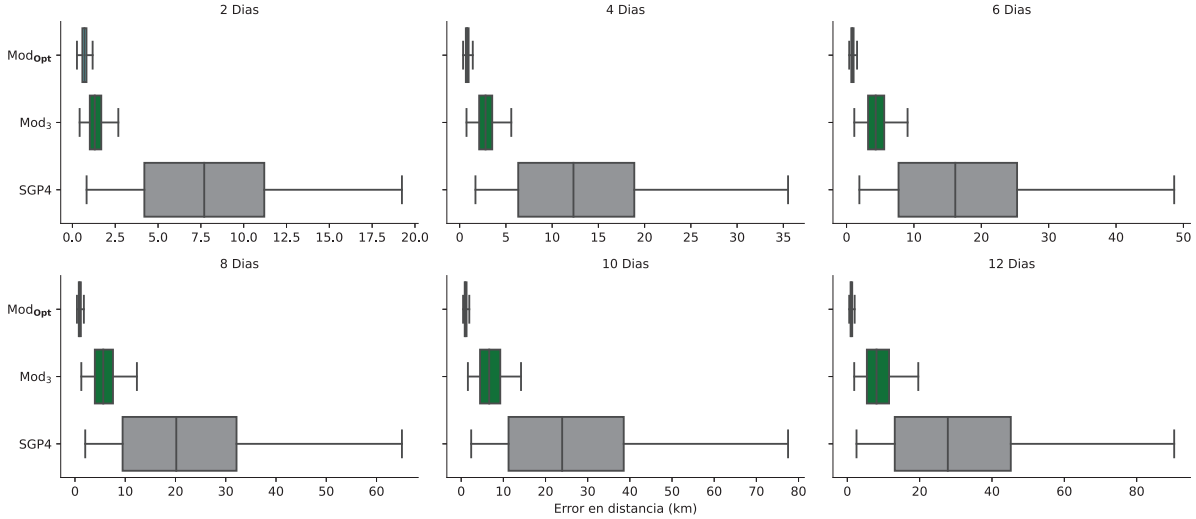


Figura 4-43.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y Mod$_3$ para las 1685 series. Las predicciones comienzan a partir del día 2.

Mod$_3$, esta diferencia se reduce a 2.26, 4.84, 7.91, 11.06, 12.60 y 17.71 km. Durante el periodo de propagación, los máximos errores cometidos por Mod$_3$ oscilan alrededor de 2.68, 5.58, 9.06, 12.33, 14.19 y 19.62 km, lo que indica una reducción del error máximo observado en SGP4 de al menos 16.91, 30.20, 39.72, 52.54, 63.65 y 70.93 km, respectivamente. En comparación con los resultados del Mod$_1$ aleatorio entrenado con una resolución de 10 minutos, los errores máximos se incrementan en 0.25, 0.17 y 0.01 los primeros 6 días, pero se reducen en 0.74, 3.61 y 3.16 km en los días restantes. Sin embargo, en el caso del modelo óptimo, se observa una diferencia de alrededor de 1.5, 4.16, 7.5, 10.56, 12.27 y 17.59 km, respectivamente. Durante esta primera fase, se registraron 32 valores atípicos el día 2, 22 el día 4, 7 el día 6 y 5 el día 12.

Análisis a partir del día 10

En la figura **4-44** se muestran los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod$_{\text{Opt}}$ y Mod$_3$ para cada uno de los 1685 TLE a partir del día 10.

La diferencia entre el bigote superior e inferior de SGP4 en los días 10, 12, 14, 16, 18 y 20 es de 78.87, 92.70, 105.53, 120.41, 134.52 y 148.16 km, respectivamente. Esta diferencia se reduce a aproximadamente 2.07, 5.38, 9.31, 18.35, 29.94 y 45.54 km, respectivamente, con el modelo Mod$_3$. Los errores máximos cometidos por este modelo son de 2.61, 6.26, 10.43, 19.47, 31.37 y 47.84 km, lo que indica una reducción en los máximos errores cometidos por SGP4 de aproximadamente 77.31, 88.68, 97.34, 103.17, 106.16 y 103.33 km, respectivamente. Aunque esta reducción es significativa en comparación con SGP4, frente al modelo óptimo, aún se conserva una diferencia de aproximadamente 0.62, 4.27, 8.44, 17.43, 29.31 y 45.76 km, respectivamente. Comparado con los resultados del Mod$_1$ aleatorio entrenado con una

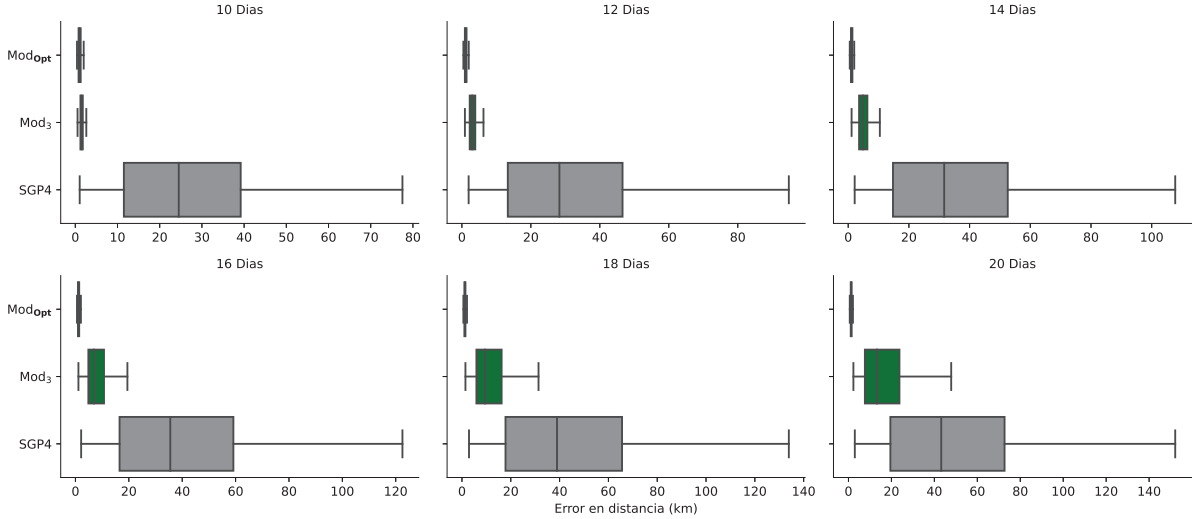


Figura 4-44.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod_{Opt} y Mod_3 para las 1685 series. Las predicciones comienzan a partir del día 10.

resolución de 10 min, los errores máximos se reducen en 0.38, 1.58 y 2.19 km los días 10, 12 y 14, respectivamente, pero se incrementan en 0.79, 5.3 y 11.52 km los días restantes. Durante este proceso, se observaron 4 valores atípicos el día 10, 5 el día 12, 4 el día 16 y 5 en los días 18 y 20.

Análisis a partir del día 12

En la figura **4-45** se muestran los diagramas de caja y bigotes que analizan los máximos errores en distancia entre AIDA y los modelos SGP4, Mod_{Opt} y Mod_3 para cada uno de los 1685 TLE a partir del día 12.

La diferencia entre los bigotes superior e inferior de SGP4 asciende aproximadamente a 92.2, 105.53, 118.84, 131.58, 143.27 y 155.18 km en los días 12, 14, 16, 18, 20 y 22, respectivamente. Sin embargo, con el modelo Mod_3, esta diferencia se reduce a alrededor de 2.41, 7.25, 14.30, 24.93, 39.91 y 60.09 km, respectivamente. Mientras que SGP4 presenta errores máximos de 90.37, 107.75, 121.06, 133.92, 149.05 y 158.58 km durante el período de propagación, el modelo Mod_3 logra reducirlos a 2.97, 8.18, 15.48, 26.85, 41.83 y 62.39 km, respectivamente. Por ejemplo, en el día 22, esta reducción alcanza al menos 95.81 km. Comparado con los errores máximos del modelo óptimo, la diferencia con Mod_3 es de aproximadamente 1.00, 6.21, 13.41, 24.81, 39.78 y 60.31 km. Al evaluar los resultados del Mod_1 aleatorio, entrenado con una resolución de 10 minutos, se observa una reducción de 0.57, 2.34 y 0.4 km en los días 12, 14 y 16, respectivamente. Sin embargo, se registra un incremento de 3.24, 8.55 y 15.5 km en los días restantes. Durante esta fase, se detectaron 3 valores atípicos el día 12, 5 el día 14, 8 los días 16 y 18, 5 el día 20 y 2 el día 22.

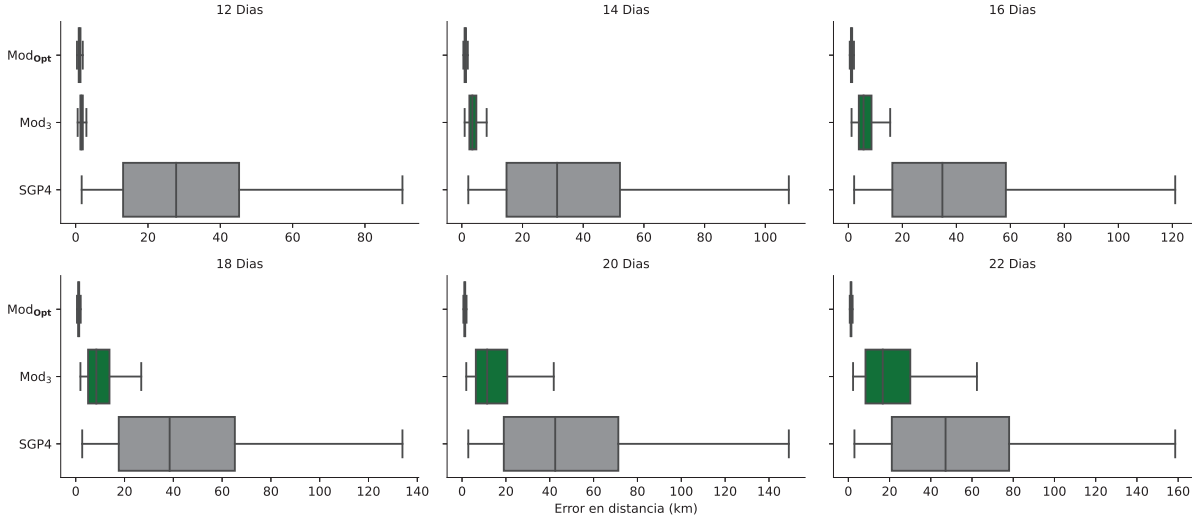


Figura 4-45.: Diagramas de caja y bigotes que muestran los máximos errores en distancia (km) entre AIDA y SGP4, Mod$_{\text{Opt}}$ y Mod$_3$ para las 1685 series. Las predicciones comienzan a partir del día 12.

Índices de mejora

En la tabla **4-51** se muestra el número de veces en las que las predicciones del Mod$_1$ empeoraron los resultados obtenidos con SGP4.

Modelo	Casos en los que $\mathcal{I}_{m_{SGP4}} > 1$ - Tiempo en días						
Mod$_3$	Día	2	4	6	8	10	12
		15	82	171	194	197	240
	Día 8	10	12	14	16	18	20
		164	140	142	157	169	154
	Día 10	12	14	16	18	20	22
		149	148	145	157	147	153

Tabla 4-51.: Número de casos en los que las predicciones del modelo Mod$_3$ empeoraron las de SGP4 desde el instante inicial, así como desde los días 8 y 10.

Durante los primeros 2 días de propagación, se identificaron 15 casos de empeoramiento. El pico máximo se registró el día 12, con 240 casos, representando un 14.24 % de las predicciones del modelo. Para este día, el índice de mejora más alto fue de 4.95 unidades, asociado a la serie sin tendencia $\varepsilon^{\theta}_{1412}$, con un incremento del error de 2.74 a 13.56 km. En las propagaciones realizadas a partir del octavo día, se registraron 164 casos durante los primeros dos días. El día 18 destacó con 169 casos, representando el 10 % de las predicciones del modelo. Para el día 20 se contabilizaron 154 casos, donde el índice más alto fue de 3.54 unidades, relacionado con la serie sin tendencia $\varepsilon^{\theta}_{857}$, con un aumento del error de 3.30 a 11.68 km. En el último periodo de propagación, el día 12 registró 149 casos que empeoraron los resultados de SGP4. El día 18, mostró el mayor número, con 157, equivalente al 9 % de las predicciones del modelo. El

día 22 se registraron 153 casos. El índice de mejora más alto para ese día es de 1.79 unidades, perteneciente nuevamente a la serie sin tendencia $\varepsilon^{\theta}_{1412}$, con un incremento del error de 2.95 a 5.28 km.

4.4. Propagador híbrido

El módulo predictivo del propagador HEncke$_{\mathbf{SGP4}}$, que se muestra en 4.1, es generado automáticamente por el entorno Python. Este programa recibe como entrada un vector que contiene el error de propagación de SGP4 del argumento de latitud θ durante las dos primeras revoluciones del satélite. Luego, el programa reconoce los patrones presentes en el vector de entrada y, basándose en esa información, propaga el comportamiento de la serie durante el horizonte temporal seleccionado por el usuario. La salida del programa es un vector que contiene las estimaciones del error cometido en las propagaciones de SGP4.

Este programa comienza importando las librerías estándar del lenguaje, así como la biblioteca de plantillas EIGEN [169], utilizada para realizar operaciones de cálculo matricial y álgebra lineal. Además, se incluye el archivo de cabecera "formatoEigen.h", el cual se encarga de transformar los datos de entrada al programa y las matrices con los coeficientes de predicción obtenidos al entrenar la RN, a un formato compatible con EIGEN.

A continuación, se definen las variables de tipo matriz de datos, donde se almacenarán los datos del vector de entrada, así como los pesos y sesgos de las matrices de la RN. Posteriormente, se lee el archivo con los datos del vector de entrada y los coeficientes predictivos para realizar los cálculos matriciales necesarios. Finalmente, se proporcionan los valores con las predicciones.

```
1
2  // Paso 1: Importar las librerias estandar de C++ y
3  //         la biblioteca de plantillas de EIGEN.
4
5  #include <iostream>
6  #include <iomanip>
7  #include <Eigen/Dense>
8  #include <fstream>
9  #include "LeerPesosdetxt.h"
10
11 // Paso 2: Definir variables.
12
13 // Matriz de datos de entrada:
14 Eigen::MatrixXd X;
15 // Matriz y sesgo de la primera capa:
16 Eigen::MatrixXd W1;
17 Eigen::MatrixXd B1;
18 // Matriz y sesgo de la segunda capa:
19 Eigen::MatrixXd W2;
20 Eigen::MatrixXd B2;
21 // Matriz y sesgo de la capa de salida:
22 Eigen::MatrixXd W3;
23 Eigen::MatrixXd B3;
24 //Matriz de Primera  multiplicación + sesgo+funcion de activación
```

```
Eigen::MatrixXd X_W1;
//Matriz de segunda  multiplicación + sesgo+funcion de activación
Eigen::MatrixXd X_W2;
// Matriz de salida, multiplicación + sesgo+funcion de activación
Eigen::MatrixXd X_W3;
// Matriz de datos de predicciones:
Eigen::MatrixXd Prediccion;

// Función principal.
int main()
{

// Paso 3: Leer el vector con los datos del error de
           propagación y los coeficientes de predicción.

  std::ifstream fin("NumeroDatos.txt");
  std::string name;
  int NumeroPredicciones, VectorEntrada;
  fin >> name >> NumeroPredicciones >> VectorEntrada ;
  X  = openData("Entrada.csv");
  // Matriz y sesgo de la primera capa:
  W1 = openData("W1.csv");
  B1 = openData("B1.csv");
  // Matriz y sesgo de la segunda capa:
  W2 = openData("W2.csv");
  B2 = openData("B2.csv");
  // Matriz y sesgo de la capa de salida:
  W3 = openData("W3.csv");
  B3 = openData("B3.csv");

// Paso 4: Realizar las respectivas multiplicaciones matriciales
           incluyendo los cálculos con las funciones de activación.

  for(int i=0;i<=NumeroPredicciones;i++)
  {
   // Primera multiplicación del vector de entrada con las matriz de pesos
     y sesgos+funcion de activación
  X_W1=(X(Eigen::all,Eigen::seq(Eigen::last+1-VectorEntrada,Eigen::last))*
   W1)+B1.transpose();
  X_W1=X_W1.unaryExpr(&tanh);

  //Segunda  multiplicación+sesgo+funcion de activación
  X_W2=(X_W1*W2)+B2.transpose();
  X_W2=X_W2.unaryExpr(&tanh);

  // Matriz de salida, multiplicación+sesgo+funcion de activación
  X_W3=(X_W2*W3)+B3.transpose();
  X_W3=X_W3.unaryExpr([](float x)
  {
  if(x>10) return 1.0*(x);
  else return 1.0*(x);});
  X.conservativeResize(X.rows(), X.cols()+1);
  X.col(X.cols()-1) << X_W3.col(0);
```

```
74   }
75
76 // Paso 5: Exportar los resultados de las estimaciones del error
                cometidos por el propagador SGP4.
77
78   for(int i=0;i<=X.cols()-1;i++)
79     {
80       if(i<X.cols())
81   {
82           std::cout <<X(i)<<std::endl;
83   }
84       else
85   {
86     std::cout <<X(i)<<";";
87   }
88
89     }
90   return 0;
91 }
```

Listing 4.1: Propagador.cpp

5. Conclusiones y trabajo futuro

Los propagadores orbitales tipo Encke emplean una combinación de métodos analíticos y numéricos para integrar las ecuaciones que gobiernan el movimiento tanto de satélites artificiales como de restos de basura espacial. Sin embargo, el uso de SGP4 en lugar del modelo de Kepler presenta dificultades debido a la complejidad para caracterizar las diferencias que deben integrarse numéricamente, dada la naturaleza particular de la construcción de SGP4.

En este trabajo se ha utilizado una metodología híbrida para caracterizar las perturbaciones no consideradas en SGP4, así como los errores inherentes a las técnicas analíticas empleadas en la construcción de las ecuaciones implementadas en este propagador. Para ello, se ha empleado un método de predicción basado en Inteligencia Artificial, específicamente las redes neuronales, junto con pseudo-observaciones generadas por un propagador numérico de alta precisión. Este nuevo propagador está especialmente diseñado para los objetos que orbitan en la región MEO del espacio, lo que hace que el modelo predictivo sea único para cualquier TLE de esta región.

Para obtener el modelo predictivo se han entrenado varias redes neuronales, variando tanto la forma de suministrar los datos de aprendizaje como la arquitectura de la red. Los datos empleados para el ajuste de las redes provienen de los satélites GSAT0203 y GALILEO–FM3 pertenecientes al sistema de posicionamiento global europeo. Con las TLE del satélite GSAT0203 se ha definido la mejor arquitectura de la red y se ha entrenado el modelo implementado en la parte híbrida del propagador. Por su parte, las TLE del satélite GALILEO–FM3 se han implementado para validar el ajuste del modelo entrenado.

Como parte de este trabajo, se ha realizado una reimplementación en Python y C++ de un entorno de trabajo que originalmente estaba desarrollado en los lenguajes de programación R y Java. Este entorno utilizaba la librería H2O para aplicar métodos de Inteligencia Artificial, así como diversas librerías estadísticas para la predicción de series temporales.

En la implementación en Python se utilizan las librerías especializadas en aprendizaje automático Tensorflow y Keras, junto con las herramientas de la biblioteca scikit-learn. Una característica destacable de esta nueva implementación es la generación automática de un programa en C++ que implementa la parte híbrida del propagador y evalúa eficientemente el modelo predictivo.

A continuación, se presentan las conclusiones y las líneas de investigación futuras que se extraen tras la realización de este trabajo.

5.1. Conclusiones

Este trabajo ha sido posible gracias al sistema de computación de altas prestaciones Beronia de la Universidad de La Rioja, el cual nos ha permitido procesar grandes volúmenes

de datos y llevar a cabo los cálculos complejos derivados de esta investigación en un tiempo razonable.

En el Capítulo 3 se han presentado dos algoritmos que implementan la metodología híbrida, uno de carácter puntual válido para una condición inicial y otro válido para una región específica del espacio. También se ha descrito la metodología a seguir para desarrollar un propagador híbrido, la cual consta de tres etapas: análisis preliminar, selección de la técnica de predicción y construcción del propagador híbrido. Asimismo, se han introducido dos índices: el primero mide la mejora del propagador híbrido respecto al propagador inicial, mientras que el segundo evalúa su capacidad de mejora respecto al propagador híbrido óptimo. Además, se ha realizado un análisis preliminar para la construcción de un propagador híbrido de tipo Encke basado en el propagador SGP4, especialmente adaptado para la región MEO. En este análisis se ha concluido que al reducir únicamente el error en el argumento de la latitud (ε^{θ}), es posible mejorar la precisión de SGP4. Esta premisa es la base sobre la cual se ha construido el propagador híbrido $\text{HEncke}_{\text{SGP4}}$.

En el Capítulo 4 se ha desarrollado el modelo predictivo, es decir, la parte híbrida del propagador $\text{HEncke}_{\text{SGP4}}$. Para ello, se ha utilizado la arquitectura inicial propuesta en [155, 156] que requiere ajustar los parámetros de la red neuronal para cada TLE. En la Sección 5.2 se ha adaptado esta arquitectura para modelar simultáneamente el comportamiento de múltiples series temporales. Además, se han evaluado dos modos de organizar el conjunto de datos utilizados para entrenar la red neuronal: el modo secuencial y el modo aleatorio. En el modo secuencial los datos se organizan de forma ordenada, mientras que en el modo aleatorio se mezclan de forma aleatoria antes de ser suministrados a la red neuronal.

Para cada uno de estos modos de organizar los datos, se han entrenado cinco modelos con la arquitectura seleccionada. En este estudio comparativo se ha concluido que cuando las muestras consecutivas pertenecen a la misma serie, es decir, en el modo secuencial, las redes tienden a converger fácilmente. Estos modelos han reducido el error en distancia hasta en 21.04 km después de 12 días de propagación. Además, se ha observado que el modelo con el mejor desempeño mejoró las predicciones de SGP4 en más de un 88 % de las veces. Sin embargo, esta disposición de las observaciones introduce un sesgo en el aprendizaje. Esto hace que el modelo reconozca fácilmente patrones vistos durante el entrenamiento; pero tenga dificultades para reproducir patrones no vistos previamente, en especial si los patrones pertenecen a un instante diferente al instante inicial de las series. En estos casos las predicciones del modelo pueden deteriorarse rápidamente a los pocos días de comenzar la propagación. De hecho, en los casos donde las propagaciones comenzaran a partir del décimo día, el porcentaje de mejora de las predicciones del modelo frente a SGP4 es solo del 37 %. Los casos donde han aparecido errores atípicos en los modelos predictivos a partir del décimo día de propagación se han separado para su posterior análisis. Se ha observado que los modelos perdían estabilidad en sus predicciones en un conjunto particular de 20 series temporales. El análisis permitió concluir que estas series se caracterizaban por pertenecer al grupo con la tendencia positiva más pronunciada, específicamente, son series cuya tendencia se ubica dentro del 16.5 % de las series con la tendencia más alta. En el caso del modo aleatorio el proceso de entrenamiento de las redes neuronales se vuelve más difícil. Sin embargo, en los casos donde el proceso converge, los modelos resultantes han sido capaces de reconocer los patrones vistos durante el entrenamiento, así como aquellos en los que patrones similares

reaparecían en diferentes instantes de tiempo. Esto ha conducido a una mejora significativa en los resultados obtenidos por SGP4 en todos los casos evaluados, con reducciones del error de $\text{HEncke}_{\text{SGP4}}$ de hasta 110 km. Además, el modelo con el mejor desempeño ha logrado mejorar los resultados de SGP4 en más del 90 % de los casos, independientemente de si los patrones habían sido vistos o no durante el proceso de entrenamiento de las redes. Por último, se observó que el tiempo de cómputo empleado por la implementación del módulo predictivo en C++ en el sistema de cómputo Beronia, aumenta aproximadamente en 1.78 segundos el tiempo requerido por el propagador SGP4.

En la Sección 5.3, se ha acometido el proceso de optimización de la arquitectura de la red neuronal desarrollada en la Sección 5.2 con el objetivo de reducir el tiempo de cálculo del modelo predictivo implementado en C++. Este proceso se ha llevado a cabo en dos etapas. En la primera etapa se ha reducido el número de puntos tomados en cada revolución de la serie, lo que resultó en una disminución del número de neuronas en la capa de entrada de la red, pasando de 84 a 12 y, posteriormente, a 6. Esta reducción condujo a una disminución en el número de coeficientes del modelo, lo que a su vez redujo el tiempo necesario para realizar una propagación, en aproximadamente, 0.84 segundos. Los resultados han indicado que, a pesar de la pérdida de información precisa del comportamiento de las series debido a la reducción de la frecuencia de muestreo, la precisión del propagador híbrido no se ha visto afectada. Esto se ha debido a que, al conservar las características generales del comportamiento de las series, el módulo predictivo del propagador ha mantenido la capacidad de reconocer y propagar los patrones de las series, independientemente del momento de selección o de la pertenencia (o no) a la etapa de entrenamiento. En la segunda etapa se ha reducido el número de neuronas de las capas ocultas de la red de 256 y 128 a 64 y 32, respectivamente. Como resultado, el tiempo de cómputo del modelo predictivo se ha acortado a, aproximadamente, 0.19 segundos en realizar una propagación; lo que representa una reducción de tiempo de más del 89 % en comparación con el módulo obtenido en la primera iteración. Este módulo ha permitido al modelo mejorar las predicciones de SGP4 en más del 85 % de las ocasiones en todos los ámbitos evaluados. Además, en esta iteración se ha logrado una reducción del error de SGP4 de hasta, aproximadamente, 100 km.

5.2. Trabajo futuro

Como ya se ha comentado, la mejora de la precisión en los sistemas de propagación orbital, sin aumentar su tiempo de cómputo, representa uno de los desafíos actuales en el ámbito del SSA. Este desafío es atribuible al rápido incremento del número de satélites y residuos espaciales que orbitan la Tierra. En este contexto, la aplicación de la metodología híbrida podría ofrecer una solución a este problema. No obstante, aún queda un extenso trabajo por realizar en esta área. A continuación se enumeran algunas ideas que han surgido durante el transcurso de esta tesis doctoral, con el objetivo de contribuir al avance futuro en el estudio de estos propagadores.

- En las pruebas realizadas el índice de mejora demostró que, aunque la precisión del propagador $\text{HEncke}_{\text{SGP4}}$ es generalmente mayor que la del SGP4, aún se encuentra considerablemente distanciada del resultado óptimo. La arquitectura final desarrollada

en esta tesis reduce significativamente el tiempo de entrenamiento del modelo, lo que facilita la inclusión de una cantidad mucho mayor de datos utilizando los mismos recursos computacionales. En este punto se propone emplear el modelo de red neuronal obtenido en este trabajo y aplicar la técnica de aprendizaje por transferencia. Esto implica llevar a cabo un proceso de reentrenamiento con una cantidad considerablemente mayor de ejemplos provenientes de diferentes satélites. El objetivo principal es permitir que el modelo capture de manera más precisa el comportamiento físico del problema.

- Durante las pruebas realizadas se observó que el modelo predictivo no tenía un buen comportamiento sobre las series sin tendencia. Esto se debió a que, en estos casos, SGP4 proporcionaba una buena aproximación, lo que dejaba poco margen de mejora para el nuevo propagador. Esta limitación se atribuyó en parte a la escasa representatividad de estas series en los datos de entrenamiento. Para abordar este problema, inicialmente, se sugiere aplicar técnicas de clasificación de series temporales para incorporar un módulo dentro del modelo. Este módulo permitiría clasificar los patrones de entrada según su tendencia antes de realizar las propagaciones. El objetivo sería que el modelo determine automáticamente si aplicar o no las correcciones del módulo predictivo a las predicciones de SGP4.

- El correcto aprendizaje de las redes neuronales depende, en gran medida, de una selección adecuada de hiperparámetros. En este trabajo la selección de hiperparámetros se basó en resultados previos. Como un trabajo futuro se propone explorar el uso de técnicas bioinspiradas para utilizar los hiperparámetros de las redes obtenidos en esta tesis y, así, conseguir una arquitectura de red con un mejor rendimiento.

- El numero de neuronas en la capa de entrada (es decir, el número de puntos por revolución) fue uno de los parámetros que ayudaron a reducir el tiempo de computación del módulo predictivo. Al disminuir la frecuencia de muestreo de las series se pierden pequeños detalles de su comportamiento; sin embargo, esto puede favorecer a la red para que reconozca con mayor facilidad las características generales de dicho comportamiento. Por este motivo, se propone incorporar una capa convolucional de dimensión 1 en el modelo, de manera similar a como se realiza en las redes convolucionales empleadas en el reconocimiento y clasificación de imágenes. Este método permitiría, mediante el uso de filtros, extraer de manera más eficiente las características del comportamiento de las series temporales; lo que facilitaría un mejor aprendizaje de la red.

- Las redes neuronales recurrentes son una clase de algoritmos de aprendizaje profundo utilizadas para procesar datos secuenciales, como las series temporales. Sin embargo, el tiempo necesario para entrenar este tipo de redes suele ser mayor que el de las redes profundas empleadas en este trabajo. Gracias a la obtención de una arquitectura de red óptima, el uso de una estructura similar a la desarrollada en este trabajo reduciría la cantidad de operaciones requeridas para el entrenamiento y la predicción de estas redes recurrentes. Por este motivo, se propone investigar la aplicación de este tipo de redes para implementar la parte híbrida de este nuevo tipo de propagadores.

- Finalmente, se propone adaptar el propagador HEncke$_{\text{SGP4}}$ para que pueda ser utilizado en otras regiones del espacio. Esta adaptación implicaría entrenar el módulo predictivo con el comportamiento de los errores de SGP4 en estas regiones.

A. Anexo: Programa informático

A.1. Librerías empleadas

Python:

- Pandas [175]. Creada para análisis interactivo de datos, convierte la información en un tipo de objeto llamado **Dataframe**, donde se organiza la información por filas y columnas. Esta herramienta es especialmente útil para la manipulación de datos reales. En el programa desarrollado, Pandas se encargó del proceso de exploración, manipulación y limpieza de los datos de entrada.

- Numpy [176]. Está orientada hacia la computación científica tradicional, enfocada en la manejo de conjuntos de datos multidimensionales. Se empleó en la manipulación de los vectores de las matrices, los cuales contienen los ejemplos suministrados a las redes neuronales durante su entrenamiento.

- Tensorflow [177]. Es un sistema de computación numérica de código libre diseñado para facilitar la implementación de las redes neuronales en la solución de problemas académicos y del mundo real. En el desarrollo de esta investigación, se empleó para construir y entrenar las redes neuronales empleadas en los experimentos.

- Matplotlib [178]. Creada para generar visualizaciones estáticas, animadas e interactivas en Python. En el entorno, se empleó para generar los informes gráficos del rendimiento del modelos entrenados, tanto en su fase de aprendizaje, como de aplicación.

- Scipy [179]. Es una librería que ofrece una amplia gama de algoritmos para optimización, integración, interpolación, problemas de valores propios, ecuaciones algebraicas, ecuaciones diferenciales, estadística y otras clases de problemas. Su uso estuvo presente en los cambios entre coordenadas satelitales (de Delaunay a cartesianas, a elementos orbitales etc).

- Multiprocessing [180]. Permite la ejecución de código de forma paralela utilizando los múltiples procesadores de un ordenador. Se empleó dentro del programa, para agilizar procesos de preprocesado de datos, cambios de coordenadas y el entrenamiento simultaneo de múltiples modelos de redes neuronales.

- Subprocess [181]. Permite generar y mantener comunicación activa con nuevos procesos de ejecución de código, a medida que se ejecuta el programa principal. En el programa desarrollado en esta tesis, se empleó para compilar, ejecutar y leer desde el lenguaje Python, el código en C++ y sus resultados.

Del código creado en C++, se destaca la biblioteca:

- Eigen [169]. Esta biblioteca está pensada para solucionar problemas de álgebra lineal: matrices, vectores, métodos de resolución numéricos y algoritmos relacionados. En el programa desarrollado, se empleó para predecir el comportamiento de las series temporales con las redes neuronales entrenadas.

A.2. Estructura del programa

El programa se compone de un archivo principal llamado **deeplearning**, un archivo secundario llamado **func_deeplearning**, y 6 módulos adicionales (*coordinates.py*, *angles.py*, *ml_misc.py*, *period.py* y *ariadna_plots.py*). El archivo **deeplearning** se encarga de crear las bases de datos para el entrenamiento y la validación de las redes neuronales, así como de generar los gráficos con los resultados obtenidos. Por otro lado, el archivo **func_deeplearning** se encarga de definir la arquitectura, entrenar la red neuronal y realizar pronósticos con las redes entrenadas.

Durante la ejecución del programa, los módulos son requeridos para realizar tareas específicas. Por ejemplo, el módulo *coordinates.py* se encarga de los cambios de coordenadas, *angles.py* normaliza magnitudes angulares, *ml_misc.py* crea los vectores para el entrenamiento y validación del aprendizaje de la red neuronal, *period.py* calcula el periodo orbital y *ariadna_plots.py* configura los parámetros para presentar los gráficos con los resultados obtenidos.

```
1                              --------------------
2                              | Deeplearning.py |
3                              --------------------
4  """
5  Entrada:
6
7     - Las carpetas Appr y Obs, con las efemérides simuladas por el
8       propagador orbital SGP4 y AIDA respectivamente.
9     - Dos archivos de control en formato "txt", nombrados como:
        general_parameters.txt y hyper_parameters.txt.
10
11          En general_parameters.txt se especifican los siguientes parámetros:
12
13           * Las efemérides contenidas en Appr y Obs, que serán empleadas en
               el entrenamiento de la red, y las que serán empleadas para
               evaluar su aprendizaje.
14
15           * El intervalo de tiempo con que se van a seleccionar los datos
               de la posición del satélite, para el entrenamiento de la red
               (resolución de muestreo).
16
17           * El numero de revoluciones del satélite que se va a emplear para
               entrenar, validar y probar la red neuronal.
18
19           * El número de redes  que  se van a entrenar.
20
```

```
21          * El método de validación de la red (validación normal o
              validación cruzada).
22
23          * La variable a modelar: r,θ,ν,R,Θ o N.
24
25          * El número de días durante el cual se va a propagar la órbita
              satelital con las redes entrenadas.
26
27          * La configuración de los gráficos con los resultados.
28
29        En hyper_parameters.txt se  especifica:
30
31          * El método para inicializar los pesos de la red.
32
33          * La forma de suministrarle los ejemplos a la red neuronal para
              su entrenamiento (estrategia aleatoria o secuencial).
34
35          * Los hiperparámetros de la red neuronal (topología, funciones de
              activación, optimizador etc.)
36
37 Salida:
38
39     - Una carpeta por cada una de las redes neuronales entrenadas. Cada
         carpeta contiene:
40
41         * Un archivo con extensión ‘‘.h5’’ y un programa ejecutable para
             linux que contiene una matriz de pesos y sesgos de la red
             neuronal entrenada.
42
43         * Una subcarpeta por cada uno de los pronósticos realizados por la
             red neuronal, donde se incluyen:
44
45                 -> El error de propagación de SGP4 frente a AIDA de la
                      variable modelada.
46
47                 -> El comportamiento del error de SGP4 frente a AIDA,
                      pronosticado por la red neuronal.
48
49                 -> La distancia medida en kilómetros, desde la posición
                      del satélite pronosticada por el SGP4, hasta la
                      posición real pronosticada por AIDA.
50
51                 -> La distancia medida en kilómetros, desde la posición
                      del satélite pronosticada por SGP4 con las correcciones
                      dadas por la red neuronal, hasta la posición real
                      pronosticada por AIDA.
52
53                 -> Una gráfica donde se superpone la serie de tiempo ϵ_t^θ
                      generada por el error en la propagación de SGP4 y el
                      pronostico dado por la red neuronal.
54
55
56
```

```
57                  -> Las gráficas que resumen el comportamiento durante el
                       entrenamiento de las métricas seleccionadas, para
                       evaluar el aprendizaje de las redes.
58
59 ------------------------------------------------------------------------
60 """
61 #   Importando las librerias y los módulos para los cálculos específicos
62
63 import sys,os
64 import json
65 import pandas as pd
66 import numpy as np
67 import tensorflow as tf
68 import period as pe
69 import angles as an
70 import ml_misc as ml_misc
71 import ariadna_plots as ap
72 import coordinates as coor
73 import func_deeplearning as fdl
74 from   distutils.util import strtobool
75 from   sys            import exit
76 from   shutil         import rmtree
77 from   itertools      import combinations
78
79 #                   Importando los archivos de control
80
81 def open_data(general_parameters.txt,hyper_parameters.txt):
82 """
83       Función diseñada para leer archivos de datos en formato ".txt".
84
85       Parámetros:
       general_parameters.txt,hyper_parameters.txt: archivos de control.
86       general_parameters.txt:especifica los parámetros para
87       configurar el entorno de trabajo.
88       hyper_parameters.txt:especifica los hiperparámetros de la red neuronal.
89
90       Retorna:
       general_param, hyper_param: Los diccionarios con las configuraciones
       del programa y los hiperparámetros de la red.
91 """
92 #                         Definiendo funciones
93
94  def data_collection(Appr,Obs):
95 """
96      Lee las efemérides simuladas por SGP4 y AIDA. Verifica que los
97      tiempos en las columnas coinciden en ambos archivos.
98
99      Parámetros:
       Appr y Obs: archivos con las efemérides.
100
101     Retorna:
       True o  False, appr, obs: un valor booleano y dos tuplas.
102    True/False:indica si los tiempos entre las efemérides coinciden.
```

```
103         False detiene el proceso de entrenamiento.
104         appr, obs: tuplas donde se almacena la información de las efemerides.
105     """
106     def switch_demo(coordinates, file.txt):
107     """
108         Convierte un archivo de texto con efemérides en coordenadas de
            delaunay, orbitales, equinoccionales, cartesianas, o variables de
            Hill a elementos orbitales. Esta función emplea el módulo
            coordinates.py.
109
110         Parámetros:
            coordinates: una cadena de texto que indica las coordenadas de la
111         matriz de entrada.
112         file.txt: matriz que contiene las efemérides en un tipo de variable.
113
114         Retorna:
115         coor: un archivo de texto con las efemerides convertidas a coordenadas
                orbitales.
116     """
117     def input_revol(kep_period,Resol):
118     """
119         Determina la resolución de muestreo de los datos de la serie
            temporal.
120
121         Parámetros:
            kep_period, Resol: el periodo kepleriano de la órbita satelital y la
122         resolución.
123
124         Retorna:
            resol: un entero que indica la frecuencia en minutos con la que se
            deben tomar dos muestras sucesivas de la serie temporal.
125     """
126     def effective_spans(kep_period):
127     """
128         Calcula  el numero de datos necesarios para entrenar y validar el
            entrenamiento de las redes neuronales.
129
130         Parámetros:
            kep_period: el periodo kepleriano de la órbita satélital.
131
132         Retorna:
            indice_1,indice_2,indice_3: indices para dividir el archivo de efemérides
            en los conjuntos de entrenamiento,
133         validación y prueba.
134     """
135     def verifications(data_train, data_valid):
136     """
137         Comprueba que la cantidad de datos solicitados por el usuario para
            entrenar la red y para validar su entrenamiento no supera la
            cantidad de datos disponibles.
138
139         Parámetros:
            data_train,data_valid: valores enteros que indican el número de datos
```

```
      requeridos para entrenar y validar la red neuronal.
140
141   Retorna:
      True o False: un valor boleano que indica si se cuenta con los datos
      suficientes para continuar con el proceso de entrenamiento de la
      red.
142 """
143 def error(Appr,Obs, var):
144 """
145   Calcula el error entre los valores propagados por SGP4 respecto a
      AIDA para la variable var indicada.
147
148   Parámetros:Obs,Obs,var: dos archivos con las efemérides obtenidas de
      SGP4 y AIDA, junto a la variable a modelar.
149   Appr: efemerides de SGP4.
150   Obs : efemerides de AIDA.
151   var : variable a modelar.
152
153   Retorna:
      error : una tupla que contiene la serie de tiempo del  error de
      propagación \epsilon_t^{var} cometido por el propagador orbital SGP4, frente a
      los resultados del propagador AIDA, en la varible var seleccionada.
154 """
155
156
157
158 def   bases_train_creator(Appr,Obs,var):
159 """
160   Crea las bases para el entrenamiento y la validación del
      entrenamiento de las redes neuronales.
161   Su ejecución se realiza en la siguiente secuencia:
162
163       - Realice un preprocesado de los datos de entrada a través de
          las funciones data_collection y switch_demo
164       - Llame la función error y guarde las salidas en el vector
          error_vect.
165       - Pase como parámetros: error_vect y las salidas de: input_revol,
          effective_spans y verifications, además de los parámetros
          indicados en general_parameters a ml_misc.py, para obtener los
          datos de entrenamiento y de validación de las redes neuronales.
166
167   - Nota: estos datos son retornados en forma de matrices de m
          columnas, donde las m-1 columnas son los predictores, la
          última columna es la variable respuesta.
168
169 #                         Ejecución del programa
170
171  - Ajuste los parámetros del experimento y lea la variable a modelar de
     general_parameters.
172
173  - Obtenga de bases_train_creator los datos de entrenamiento y validación de
     la red neuronal.
174
```

```
 - Envíe como parámetros los datos de entrenamiento y validación  a las
   funciones en FuncDeepLearning, FuncDeepLearning para obtener:

       * Los resultados de las predicciones hechas por cada una de las
         redes entrenadas, sobre el comportamiento de la variable
         seleccionada.

       * El error en km en la propagación de la órbita de SGP4 frente a
         AIDA.

       * El error en km en la propagación de la órbita de SGP4 frente a
         AIDA, una vez en SGP4 se halla corregido con las predicciones de
         la red neuronal, los valores de la variable seleccionada.

       * El error en km en la propagación de la órbita de SGP4 frente a
         AIDA, una vez en SGP4 se halla reemplazado los valores de la
         variable seleccionada con los valores reales dados por AIDA.

       * Genere los gráficos donde se detalle el comportamiento real del
         error cometido por SGP4 frente a AIDA en la variable
         seleccionada.

       * Guarde los resultados obtenidos.
"""
```

Listing A.1: Deeplearning.py

```
                          ------------------------
                          | FuncDeeplearning.py |
                          ------------------------
#                           Importando librerias

from   sklearn.model_selection  import KFold
from   tensorflow.keras.callbacks import EarlyStopping
import os
import pandas as pd
import numpy as np
import Coordinates as coor
import tensorflow  as tf
from   distutils.util import strtobool
from   matplotlib import pyplot
import shutil
import subprocess

def open_data(hyper_parameters.txt):
"""
      Función diseñada para leer archivos de datos en formato ".txt".

      Parámetros:
      hyper_parameters.txt:especifica los hiperparámetros de la red neuronal.

      Retorna:
      hyper_param: diccionario con las configuraciones del programa y los
```

```
27        hiperparámetros de la red.
28 """
29
30 #                          Definiendo funciones
31
32 def optimizer_and_metrics(opt,lr,ls):
33 """
34     Configura la arquitectura de la red neuronal
35
36     Parámetros:
37     opt, lr, ls: optimizador, tasa de aprendizaje y la función de error,
       de acuerdo a las instrucciones dadas en hyper_parameters.
38
39     Retorna:
40     opt_metric: una lista con tipos de datos que guardan la información del
                   optimizador, la tasa de aprendizaje y la función de error
                   seleccionada.
41 """
42
43 def data(train_vect,valid_vect):
44 """
45     Realiza una partición supervisada de los datos de entrenamiento y
       validación empleando el método de la ventana deslizante.
46
47     Parámetros:
48     train_vect, valid_vect: listas con los datos seleccionados para entrenar
       y validar el entrenamiento de la red neuronal.
49
50     Retorna:
51     train_x, valid_x, train_y, valid_y:  dos matrices con los vectores de
       entrada, y dos vectores con las salidas esperadas de la red neuronal.
52 """
53 def data_cross_validation(train_vect,valid_vect):
54 """
55     Divide los datos en bloques para la validación cruzada
56
57     Parámetros:
58     train_vect, valid_vect: listas con los datos seleccionados para entrenar
       y validar el entrenamiento de la red neuronal.
59
60     Retorna:
61     base_training: un arreglo que contiene los datos preparados para el
       entrenamiento de la red, pero particionados en bloques.
62 """
63 def neuronal_net(layer_neur_num,win,activ_funct,layers):
64 """
65     Crea la red neuronal aplicando las funciones de la libreria
       tensorflow y keras.
66
67     Parámetros:
68     layers,win,layer_neur_num,activ_funct: número de capas, numero de neuronas
       de la primera capa, número de neuronas de las capas ocultas y
       funciones de acitivación.
```

```
69
70      Los pasos se muestran a continuación:
71
72          - Defina el tipo de conexión de la red.
73              model=tf.keras.models.Sequential()
74          - Indique el algoritmo para inicializar los pesos de la red.
75              initializer=tf.keras.initializers.GlorotNormal()
76          - Construya la arquitectura de la red, con el número de neuronas
              solicitado, establezca las funciones de activación y el
              algoritmo de inicialización de pesos indicado por el usuario.
              model.add(LayerNeurNum,win,ActivFunct,initializer)
77
78      Retorna:
79      model: una arquitectura de red neuronal lista para ser entrenada.
80  """
81  def save(model,path)
82  """
83      Guarda la red neuronal entrenada en un archivo con extensión ".h5" y
        la información relevante del proceso de entrenamiento, como el número
        de épocas, el mínimo error de ajuste alcanzado, el valor del lr entre
        otros.
84
85      Parámetros:
86      model,path= el modelo de red entrenado y la ruta donde se guarda.
87
88      Retorna:
89      True o False: un valor booleano que indica si el proceso se realizó con
        éxito.
90  """
91  def  check(models_num,general_parameters)
92  """
93      función que comprueba que los modelos entrenados models_num son todos
        los modelo solicitados en general_parameters.
94  """
95  #                        Ejecución del programa
96  """
97      Lea los hiperparámetros para la arquitectura de la red, definidos en
        hyper_param.
98
99      El programa se ejecuta a través de los siguientes pasos:
100
101     Llame la función optimizer_and_metrics para configurar la arquitectura  de
        la red.
102
103     Construya la red neuronal, utilizando la función neuronal_net().
104
105     Compile la red usando el comando model.compile().
106
107     Entrene la red neuronal con el comando model.fit().
108     Guarde el modelo entrenado usando la función save().
109
110     Verifique que se han entrenado el número de modelos solicitados usando
        la función check(). Si se cumple continue con la construcción del
```

```
111     modelo híbrido, si nó, entrene las redes faltantes.
112 """
```

Listing A.2: FuncDeeplearning.py

```
1                               ------------------------
2                               |    coordinates.py    |
3                               ------------------------
4 #                         Importando librerias y módulos
5
6 from   scipy.optimize import fsolve
7 import warnings
8 import pandas as pd
9 import numpy as np
10 import CodigosPython.H_Angles as A
11 import multiprocessing as mp
12 from   sys import exit
13
14 #                         Definiendo funciones
15
16 def cart_anal2hyb(anal,fore):
17 """
18     Convierte coordenadas cartesianas  (analíticas + predicción del error)
       en híbridas.
19     Parámetros:
20     anal, fore: matrices que contienen las coordenadas cartesianas.
21     anal: modelo analítico.
22     fore: predicción del error.
23
24     Retorna:
25
26     hyb: matriz con las coordenadas cartesianas del modelo híbrido.
27 """
28 def cart2dist(cart_true,cart_test):
29 """
30     Convierte 2 conjuntos de coordenadas cartesianas a errores en
       distancia.
31
32     Parámetros:
33     cart_true,cart_test: matrices que contienen las coordenadas cartesianas.
34     cart_true: coordenadas precisas.
35     cart_test: coordenadas del modelo evaluado.
36
37     Retorna:
38     er_dist:   matriz con los errores en distancia.
39 """
40 def cart2dln(cart):
41 """
42     Convierte coordenadas cartesianas a variables de delaunay.
43
44     Parámetros:
45     cart: matriz que contiene las coordenadas cartesianas.
46
```

```
47      Rertorna:
48      dln: matriz que contiene las variables de Delaunay
49  """
50  def cart2equi(cart):
51  """
52      Convierte coordenadas cartesianas a elementos equinocciales.
53
54      Parámetros:
55      cart: matriz que contiene las coordenadas cartesianas.
56
57      Retorna:
58      equi: matriz con los elementos equinocciales.
59  """
60  def cart2hill(cart):
61  """
62      Convierte coordenadas cartesianas a variables de hill
63      (polares-nodales).
64      Parámetros:
65      cart: matriz que contiene las coordenadas cartesianas.
66      Retorna:
67      hill: matriz que contiene las variables de Hill.
68  """
69  def cart2orb(cart):
70  """
71      Convierte coordenadas cartesianas a elementos orbitales.
72
73      Parámetros:
74      cart: matriz que contiene las coordenadas cartesianas.
75
76      Retorna:
77      orb : matriz con los elementos orbitales.
78  """
79  def dln_anal2hyb(anal,fore):
80  """
81      Convierte las variables de Delaunay (analitico + predic. error) en
82      híbridas.
83
84      Parámetros:
85      anal, fore: matrices que contienen las variables de Delaunay.
86      anal: modelo analítico.
87      fore: predicción del error.
88
89      Retorna:
90      hyb: matriz con las variables de Delaunay del modelo híbrido.
91  """
92  def dln2cart(dln):
93  """
94      Convierte variables de delaunay a coordenadas cartesianas.
95
96      Parámetros:
97      dln: matriz que contiene las variables de Delaunay.
98
99      Retorna:
```

```
100     cart: matriz con las coordenadas cartesianas.
101 """
102 def dln2dist(dln_true,dln_test):
103 """
104     Convierte 2 conjuntos de variables de delaunay a errores en distancia.
105 
106     Parámetros:
107     dln_true, dln_test: matrices que contienen las variables de Delaunay.
108     dln_true: variables precisas.
109     dln_test: variables del modelo evaluado.
110 
111     Retorna:
112     er_dist: matriz con los errores en distancia.
113 """
114 def dln2equi(dln):
115 """
116    Convierte variables de Delaunay a elementos equinocciales.
117 
118    Parámetros:
119    dln: matriz que contiene las variables de Delaunay.
120 
121    Retorna:
122    equi: matriz con los elementos equinocciales.
123 """
124 def dln2hill(dln):
125 """
126     Convierte variables de Delaunay a variables de Hill (polares-nodales).
127 
128     Parámetros:
129     dln: matriz que contiene las variables de Delaunay.
130 
131     Retorna:
132     hill: matriz con las variables de Hill.
133 """
134 def dln2orb(dln):
135 """
136     Convierte variables de delaunay a elementos orbitales.
137 
138     Parámetros:
139     dln: matriz que contiene las variables de Delaunay.
140 
141     Retorna:
142     orb: matriz con los elementos orbitales.
143 """
144 def equi_anal2hyb(anal,fore):
145 """
146     Convierte elementos equinocciales (analítico + predic. error) a
147     híbridos.
148 
149     Parámetros:
150     anal, fore: matrices que contienen los elementos equinocciales.
151     anal: modelo analítico.
152     fore: predicción del error.
```

```
153
154     Retorna:
155     hyb: matriz con los elementos equinocciales del modelo híbrido.
156 """
157 def equi2cart(equi):
158 """
159     Convierte elementos equinocciales a coordenadas cartesianas.
160
161     Parámetros:
162     equi: matriz que contiene los elementos equinocciales.
163
164     Retorna:
165     cart: matriz con las coordenadas cartesianas.
166 """
167 def equi2dist(equi_true, equi_test):
168 """
169     Convierte 2 sets elementos equinocciales a errores en distancia.
170
171     Parámetros:
172     equi_true, equi_test: matrices que contienen los elementos equinocciales.
173     equi_true: variables precisas
174     equi_test: variables del modelo evaluado.
175
176     Retorna:
177     er_dist: matriz con los errores en distancia
178 """
179 def equi2dln(equi):
180 """
181     Convierte elementos equinocciales a variables de delaunay.
182
183     Parámetros:
184     equi: matriz que contiene los elementos equinocciales.
185
186     Retorna:
187     dln: matriz con las variables de Delaunay.
188 """
189 def equi2hill(equi):
190 """
191     Convierte elementos equinocciales a variables de hill
192     (polares-nodales).
193
194     Parámetros:
195     equi: matriz que contiene los elementos equinocciales.
196
197     Retorna:
198     hill: matriz con las variables de Hill.
199 """
200 def equi2orb(equi):
201 """
202     Convierte elementos equinocciales a elementos orbitales.
203
204     Parámetros:
205     equi: matriz que contiene los elementos equinocciales.
```

```
206
207     Retorna:
208     orb: matriz con los elementos orbitales.
209 """
210 def hill_anal2hyb(anal,fore):
211 """
212     Convierte variables de Hill (analítico + predic. error) a híbridas.
213
214     Parámetros:
215     anal, fore: matrices que contienen las variables de Hill.
216     anal: modelo analítico.
217     fore: predicción del error.
218
219     Retorna:
220     hyb: matriz con las variables de Hill del modelo híbrido
221 """
222 def hill2cart(hill):
223 """
224     Convierte variables de Hill (polares-nodales) a coordenadas
225     cartesianas.
226
227     Parámetros:
228     hill: matriz que contiene las variables de Hill.
229
230     Retorna:
231     cart: matriz con las coordenadas cartesianas.
232 """
233 def hill2dist(hill_true, hill_test):
234 """
235     Convierte 2 conjuntos de variables de Hill (polares-nodales) a errores
        en distancia.
236
237     Parámetros:
238     hill_true, hill_test: matrices que contienen las variables de Hill.
239     hill_true: variables precisas.
240     hill_test: variables del modelo evaluado.
241
242     Retorna:
243     er_dist: matriz con los errores en distancia.
244 """
245 def hill2dln(hill):
246 """
247     Convierte variables de Hill (polares-nodales) a variables de Delaunay.
248
249     Parámetros:
250     hill: matriz que contiene las variables de Hill.
251
252     Retorna:
253     dln: matriz con las variables de Delaunay.
254 """
255 def hill2equi(hill):
256 """
257     Convierte variables de Hill (polares-nodales) a elementos
```

```
258     equinocciales.
259
260     Parámetros:
261     hill: matriz que contiene las variables de Hill.
262
263     Retorna:
264     equi: matriz con los elementos equinocciales.
265 """
266 def hill2orb(hill):
267 """
268     Convierte variables de Hill (polares-nodales) a elementos orbitales.
269
270     Parámetros:
271     hill: matriz que contiene las variables de Hill.
272
273     Retorna:
274     orb: matriz con los elementos orbitales.
275 """
276 def orb_anal2hyb(anal, fore):
277 """
278     Convierte elementos orbitales (analítico + predic. error) a híbridos.
279
280     Parámetros:
281     anal, fore: matrices que contienen los elementos orbitales.
282     anal: modelo analítico.
283     fore: predicción del error.
284
285     Retorna:
286     hyb: matriz con los elementos orbitales del modelo híbrido.
287 """
288 def orb2cart(orb):
289 """
290     Convierte elementos orbitales a coordenadas cartesianas.
291
292     Parámetros:
293     orb: matriz que contiene los elementos orbitales.
294
295     Retorna:
296     cart: matriz con las coordenadas cartesianas.
297 """
298 def orb2dist(orb_true, orb_test):
299 """
300     Convierte 2 conjuntos de elementos orbitales a errores distancia.
301
302     Parámetros:
303     orb_true, orb_test: matrices que contienen los elementos orbitales.
304     orb_true: elementos precisos.
305     orb_test: elementos del modelo evaluado.
306
307     Retorna:
308     er_dist: matriz con los errores en distancia.
309 """
310 def orb2dln(orb):
```

```
"""
    Convierte elementos orbitales a variables de Delaunay.

    Parámetros:
    orb: matriz que contiene los elementos orbitales.

    Retorna:
    dln: matriz con las variables de Delaunay.
"""
def orb2equi(orb):
"""
    Convierte elementos orbitales a elementos equinocciales.

    Parámetros:
    orb: matriz que contiene los elementos orbitales.

    Retorna:
    equi: matriz que recogerá los elementos equinocciales.
"""
def orb2hill(orb):
"""
    Convierte elementos orbitales a variables de Hill (polares-nodales).

    Parámetros:
    orb: matriz que contiene los elementos orbitales.

    Retorna:
    hill: matriz con las variables de Hill.
"""
```

Listing A.3: coordinates.py

```
                            ------------------------
                            |       Angles.py      |
                            ------------------------
#                       Importando la libreria necesaria
import numpy as np

#                             Definiendo funciones
def rem(x,y):
"""
    Devuelve el resto después de dividir a por b, donde a es el dividendo y
    b es el divisor. Esta función suele recibir el nombre de operación
    de resto. La función rem sigue la convención de que rem(a,0) es Inf.

    Parámetros:
    x: dividendo.
    y: divisor.

    Retorna:
    out: resto con signo de la división x/y.
"""
def rem2pi_sym(x):
```

```
20 """
21     Normaliza magnitudes angulares al intervalo (-pi,pi].
22 
23     Parámetros:
24     x:datos de partida.
25 
26     Retorna:
27     out:datos normalizados.
28 """
29 def rem2pi(x):
30 """
31     Normaliza magnitudes angulares al intervalo [0,2pi).
32     La función puede ser aplicada a cualquier matriz o vector.
33     No aplicar a matrices que incluyan magnitudes no angulares, pues
       modifica cualquier valor no comprendido en [0,2pi).
34 
35     Parámetros:
36     x:datos de partida.
37 
38     Retorna:
39     out:datos normalizados.
40 """
41 def remove2pi(x,holgura)
42 """
43     Transforma valores próximos a +/-(2*pi) a valores próximos a $0$.
44 
45     Parámetros:
46     x:datos de partida.
47     holgura: todos los valores comprendidos entre +/-(2*pi)-holgura y
       +/-(2*pi)+holgura serán transformados restándoles/sumándoles 2*pi.
48 
49     Retorna:
50     out:datos transformados.
51 """
52 def rem360_sym(x):
53 """
54     Normaliza magnitudes angulares al intervalo (-180,180].
55     La función puede ser aplicada a cualquier matriz o vector. No
56     aplicar a matrices que incluyan magnitudes no angulares, pues modifica
       cualquier valor no comprendido en (-180,180].
57 
58     Parámetros:
59     x:datos de partida.
60 
61     Retorna:
62     out:datos normalizados.
63 """
64 
65 def rem360(x):
66 """
67     Normaliza magnitudes angulares al intervalo [0,360).
68 
69     Parámetros:
```

```
    x:datos de partida.
    Retorna:
    out:datos normalizados.
"""

def remove360(x, holgura):
"""
    Transforma valores próximos a +/-(360) a valores próximos a 0.
    La función puede ser aplicada a cualquier matriz o vector.

    Parámetros:
    x:datos de partida.
    holgura: todos los valores comprendidos entre +/-(360)-holgura y
   +/-(360)+holgura serán transformados restándoles/sumándoles 360.

    Retorna:
    out:datos transformados.
"""
```

Listing A.4: Angles.py

```
#                       Importando la libreria necesaria
import numpy as np
def windowed_dataset(ephem,column,num_inputs,num_outputs,row_initial,
    row_final,sampling_period,vect_step)
"""
    Genera vectores de entrenamiento y validación a partir de las matrices
    de efemérides usando la ventana deslizante.

    Parámetros:
    ephem: matriz de efemérides (filas: efemérides; columnas: variables).
    column:columna de la variable a extraer (número o nombre de la
          columna).
    num_inputs :número de entradas que tendrá el algoritmo predictivo.
    num_outputs:número de salidas que tendrá el algoritmo predictivo
              (Valor por defecto: 1).
    row_initial:fila de la efeméride desde la que se desea comenzar a
              generar los vectores (valor por defecto: 1).
    row_final_ :fila de la efeméride hasta la que se desea generar los
              vectores (valor por defecto: última fila, nrow(ephem)).
    sampling_period:periodo de muestreo. Permite reducir la resolución de
                 matrices de efemérides muy densas.
    vect_step:paso entre cada dos vectores a generar (número de posiciones
             que debe avanzar la ventana deslizante entre cada dos
              vectores).

    Retorna:
    vect:matriz de vectores (filas: vectores; columnas: entradas,salidas).
"""
```

Listing A.5: ml_misc.py

```
                            ------------------------
                            |       period.py      |
                            ------------------------
#                         Importando la libreria necesaria
import numpy as np

#                             Definiendo funciones

def a2P(a):
"""
    Convierte Semieje mayor a Periodo kepleriano

    Parámetros:
    a: semieje mayor [km].

    Retorna:
    P: periodo kepleriano [s].
"""
def P2a(P):
"""
    Convierte  Periodo kepleriano a Semieje mayor.

    Parámetros:
    P: periodo kepleriano [s].

    Retorna:
    a: semieje mayor [km].
"""
```

Listing A.6: period.py

```
                            ------------------------
                            |  Ariadna\_plots.py   |
                            ------------------------
#                         Importando la libreria necesaria
import pandas as pd
from matplotlib import pyplot

#                             Definiendo funciones
def plot_fore(fore_seq,fore_real,path_plot,train_seq,valid_seq,
variable):
"""
    Parámetros:
    fore_seq: vector con los datos del pronostico de la red.
    fore_real: vector con los datos reales del eror de propagación.
    path_plot: ruta donde se almacena el modelo.
    variable: variable modelada (se emplea para generar los títulos de la
    gráfica, y las etiquetas de los ejes.)

```

```
18      Retorna:
19      Un gráfico donde se muestra dos series de tiempo, la primera muestra
        el comportamiento real del error del propagador base, la segunda, el
        comportamiento del error modelado por el propagador Encke-híbrido.
20  """
```

Listing A.7: Ariadna_plots.py

B. Anexo: Funciones de error o coste

Frecuentemente, en redes neuronales, se emplea como función de coste el error cuadrático medio (MSE) aplicado sobre los N ejemplos del conjunto de datos de entrenamiento:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x) - y(x))^2,$$

donde $f(x)$ e $y(x)$ indican las salidas de la red y las salidas deseadas para cada uno de los ejemplos $x \in N$. En ocasiones, se calcula la raíz cuadrada del MSE con el fin de que la medida esté en las mismas unidades que los datos de entrada:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x) - y(x))^2}.$$

El problema de usar errores al cuadrado es la sensibilidad de las medidas a la presencia de datos anómalos, también conocidos como *outliers* en inglés, lo que puede provocar que algunos valores de las medidas del error sean desproporcionados. Por este motivo, suele emplearse el error absoluto medio (MAE), definido como:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|(f(x) - y(x))|.$$

Esta medida representa la diferencia en valor absoluto entre las predicciones $f(x)$ y las observaciones $y(x)$. El hecho de que la contribución directa de cada error individual sea proporcional al valor absoluto del error hace que esta medida sea más robusta frente a los valores atípicos.

La medida normalizada del MAE es el error porcentual absoluto medio, conocido como MAPE por sus siglas en inglés, definido como:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{|(f(x) - y(x))|}{|y(x)|}.$$

Esta medida proporciona el error medio cometido en la estimación de $f(x)$ expresado como un porcentaje del valor $y(x)$. Es útil para evaluar el rendimiento de un modelo en términos de precisión relativa.

De acuerdo con Rob Hyndman, un investigador reconocido por sus trabajos en series temporales, es recomendable utilizar la función de error MAPE para evaluar el desempeño de los modelos creados para la predicción de series temporales [182]. Alternativamente, propone emplear el error medio absoluto escalado (MASE) por sus siglas en inglés. MASE es una

medida que permite, entre otros aspectos, comparar predicciones sobre diferentes conjuntos de datos, penalizar de igual manera errores positivos y negativos y tener una fácil interpretación. Estas medidas de error están integradas en la mayoría de software estadístico y librerías dedicadas al Machine Learning, como R [158], Scipy [179], Tensorflow [177] y Pytorch [183].

Bibliografía

[1] J.N. Pelton, S. Madry, and S. Camacho-Lara. *Handbook of Satellite Applications*. Springer New York, 2018.

[2] A. Froehlich. *Legal Aspects Around Satellite Constellations: Volume 2*. Studies in Space Policy. Springer International Publishing, 2021.

[3] E.D. Kaplan and C. Hegarty. *Understanding GPS/GNSS: Principles and Applications, Third Edition*. GNSS Technology and Applications Series. Artech House Publishers, 2017.

[4] S. Tan. *GNSS Systems and Engineering: The Chinese Beidou Navigation and Position Location Satellite*. Wiley, 2018.

[5] Z. Dragos. *Galileo European Satellite Navigation System*. Wiley, 2020.

[6] N. Pettorelli. *Satellite Remote Sensing and the Management of Natural Resources*. OUP Oxford, 2019.

[7] V. Levizzani, C. Kidd, D.B. Kirschbaum, C.D. Kummerow, K. Nakamura, and F.J. Turk. *Satellite Precipitation Measurement: Volume 1*. Advances in Global Change Research. Springer International Publishing, 2020.

[8] T. Sgobba, G.E. Musgrave, G. Johnson, and M.T. Kezirian. *Safety Design for Space Systems*. Elsevier Science, 2023.

[9] M. Madi and O. Sokolova. *Space Debris Peril: Pathways to Opportunities*. CRC Press, 2020.

[10] H. Klinkrad. *Space Debris: Models and Risk Analysis*. Springer Praxis Books. Springer Berlin Heidelberg, 2006.

[11] John Danby. Fundamentals of celestial mechanics. *Richmond: Willman-Bell*, 1992.

[12] O. Montenbruck and E. Gill. *Satellite Orbits: Models, Methods and Applications*. Springer Berlin Heidelberg, 2012.

[13] David A Vallado. *Fundamentals of astrodynamics and applications*, volume 12. Springer Science & Business Media, 2001.

[14] Dirk. Brouwer. Solution of the problem of artificial satellite theory without drag. *The Astronomical Journal*, 64:378, 1959.

[15] Kaare Aksnes. A Second-Order Artificial Satellite Theory Based on an Intermediate Orbit. *The Astronomical Journal*, 75:1066, November 1970.

[16] R. H. Lyddane. Small eccentricities or inclinations in the Brouwer theory of the artificial satellite. *The Astronomical Journal*, 68:555, October 1963.

[17] A.C. Long, R. Pajerski, R. Luczak, United States. National Aeronautics, Space Administration, and Goddard Space Flight Center. *Goddard Trajectory Determination System (GTDS): Mathematical Theory*. Goddard Space Flight Center, 1989.

[18] Jesús Peláez Álvarez, José Manuel Hedo Rodríguez, and Pedro Rodriguez de Andrés. A special perturbation method in orbital dynamics. *Celestial Mechanics and Dynamical Astronomy*, 97(2):131–150, Febrero 2007.

[19] Jerome Vetter. Fifty years of orbit determination: Development of modern astrodynamics methods. *J. Hopkins Appl. Tech. D*, 27, 01 2007.

[20] Paul Cefola, Donald Phillion, and K.S. Kim. Improving access to the semi-analytical satellite theory. *Presented at the AAS/AIAA Astrodynamics Specialist Conference, Pittsburgh, PA*, pages 09–341, 01 2009.

[21] J. J. F. Liu and R. L. Alford. Semianalytic Theory for a Close-Earth Artificial Satellite. *Journal of Guidance Control Dynamics*, 3(4):304–311, July 1980.

[22] J.G. Neelon, Paul Cefola, and R.J. Proulx. Current development of the draper semianalytical satellite theory standalone orbit propagator package. *J. Astronaut. Sci.*, 97:97–731, 01 1997.

[23] Ivan Perez, Juan San Juan, Montserrat San-Martín, and Luis Lopez. Application of computational intelligence in order to develop hybrid orbit propagation methods. *Mathematical Problems in Engineering*, 2013, 01 2013.

[24] Juan Félix San-Juan, Montserrat San-Martín, and Iván Pérez. An economic hybrid J_2 analytical orbit propagator program based on SARIMA models. *Mathematical Problems in Engineering*, 2012:15 pages, 2012. Article ID 207381.

[25] JF San-Juan, M San-Martín, and I Pérez. Hybrid perturbation methods: modelling the j2 effect through the kepler problem. In *Advances in the Astronautical Sciences. Spaceflight mechanics 2015: proceedings of the 25th AAS/AIAA Space Flight Mechanics Meeting*, volume 155, pages 3031–3046, 2015.

[26] Oliver. Montenbruck and Eberhard. Gill. *Satellite Orbits Models, Methods and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1st ed. 2000. edition, 2000.

[27] Herbert Goldstein. *Classical Mechanics*. Addison-Wesley, 1980.

[28] Richard H. Battin. *An introduction to the mathematics and methods of astrodynamics*. AIAA education series. American Institute of Aeronautics and Astronautics, New York, N.Y, 1987.

[29] https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits.

[30] https://earthobservatory.nasa.gov/features/OrbitsCatalog.

[31] R.E. Sheriff and Y.F. Hu. *Mobile Satellite Communication Networks*. Online access: EBSCO Computers & Applied Sciences Complete. Wiley, 2001.

[32] W.M. Kaula. *Theory of Satellite Geodesy: Applications of Satellites to Geodesy*. Dover Earth Science. Dover Publications, 2013.

[33] Stefano Casotto. *Nominal ocean tide models for TOPEX precise orbit determination*. The University of Texas at Austin, 1989.

[34] W. D. Mcclain. A recursively formulated first-order semianalytic artificial satellite theory based on the generalized method of averaging. volume 1: The generalized method of averaging applied to the artificial satellite problem. 1977.

[35] G.T. Tseng, American Astronautical Society, American Institute of Aeronautics, and Astronautics. *Astrodynamics, 1983: Proceedings of the AAS/AIAA Astrodynamics Conference Held August 22-25, 1983, Lake Placid, New York*. Advances in astronautical sciences. American Astronautical Society, 1984.

[36] P. Gurfil and P.K. Seidelmann. *Celestial Mechanics and Astrodynamics: Theory and Practice*. Astrophysics and Space Science Library. Springer Berlin Heidelberg, 2016.

[37] Chia-Chun George Chao. *Applied Orbit Perturbation and Maintenance*. The Aerospace Press, American Institute of Aeronautics and Astronautics, Reston, VA, USA, 2005.

[38] Mirco Rasotto, Pierluigi Di Lizia, Alessandro Morselli, Roberto Armellin, Alexander Wittig, Celia Yabar, Guillermo Ortega, and Mauro Massari. Differential Algebra Space Toolbox for nonlinear uncertainty propagation in Space Dynamics. In *Proceedings 6th International Conference on Astrodynamics Tools and Techniques, ICATT 2016*, Darmstadt, Germany, March 2016. European Space Agency (ESA).

[39] M. Berz. The method of power series tracking for the mathematical description of beam dynamics. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 258(3):431–436, August 1987.

[40] J. R. Dormand and P. J. Prince. Practical Runge-Kutta processes. *SIAM Journal on Scientific and Statistical Computing*, 10(5):977–989, 1989.

[41] H. Bolandi, M. H. Ashtari Larki, M. Abedi, and M. Esmailzade. Gps based onboard orbit determination system providing fault management features for a leo satellite. *The Journal of Navigation*, 66(4):539–559, 2013.

[42] Yoshihide Kozai. The motion of a close earth satellite. *The Astronomical Journal*, 64:367, 1959.

[43] Y. Kozai. Second-order solution of artificial satellite theory without air drag. *The Astronomical Journal*, 67:446, 1962.

[44] Felix R Hoots. Spacetrack report no. 3, models for propagation of norad element sets. *http://www. itc. nl/-bakker/orbit. html*, 1980.

[45] David Vallado, Paul Crawford, Ricahrd Hujsak, and TS Kelso. Revisiting spacetrack report# 3. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6753, 2006.

[46] Felix Roach Hoots, Paul W. Schumacher, and Robert Glover. History of analytical orbit modeling in the u.s. space surveillance system. *Journal of Guidance Control and Dynamics*, 27:174–185, 2004.

[47] High-Level Expert Group on Artificial Intelligence. A definition of ai: Main capabilities and scientific disciplines. Technical report, EUROPEAN COMMISSION, 2019.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[49] David Levy, editor. *Computer Chess Compendium*. Springer-Verlag, Berlin, Heidelberg, 1988.

[50] Haroon Sheikh, Corien Prins, and Erik Schrijvers. *Artificial Intelligence: Definition and Background*, pages 15–41. Springer International Publishing, Cham, 2023.

[51] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[52] F. Berzal and F.B. Galiano. *Redes Neuronales and Deep Learning - Volumen 1 y 2: Regularización, Optimización Y Arquitecturas Especializadas [formato 8. 5 X 11]*. Independently Published, 2019.

[53] L. Graesser and W.L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley Data & Analytics Series. Pearson Education, 2019.

[54] R.S. Sutton and A.G. Barto. *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[55] A. Courville Y. Bengio and P. Vincent. representation learning: A review and new perspectives". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828, 2013.

[56] Agre G Zlatareva D Hadjidekov V Lazarov N. Toneva D, Nikolova S. Machine learning approaches for sex estimation using cranial measurements. *International journal of legal medicine*, 135(3):951–966, 2021.

[57] Maria Tzelepi, Paraskevi Nousi, Nikolaos Passalis, and Anastasios Tefas. Chapter 10 - representation learning and retrieval. In Alexandros Iosifidis and Anastasios Tefas, editors, *Deep Learning for Robot Perception and Cognition*, pages 221–241. Academic Press, 2022.

[58] Y LeCun. *Connexionist learning models*. PhD thesis, PhD thesis, Universite Pierre et Marie Curie (Paris), 1987.

[59] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[60] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[61] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, page 4–11, New York, NY, USA, 2014. Association for Computing Machinery.

[62] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018.

[63] Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[64] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[65] Joseph M Hilbe. *Logistic regression models*. CRC press, 2009.

[66] Derek A. Pisner and David M. Schnyer. Chapter 6 - support vector machine. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 101–121. Academic Press, 2020.

[67] Roger J Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, volume 14. Citeseer, 2000.

[68] Gordon K Smyth. Nonlinear regression. *Encyclopedia of environmetrics*, 3:1405–1411, 2002.

[69] X. Wang, Y.R. Yue, and J.J. Faraway. *Chapter 6-Bayesian Regression Modeling with INLA*. Chapman & Hall/CRC Computer Science & Data Analysis. CRC Press, 2018.

[70] George AF Seber and Alan J Lee. Polynomial regression. *Linear Regression Analysis*, pages 165–185, 2003.

[71] Doriana Marilena D'Addona. *Neural Network*, pages 911–918. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[72] Melody Y. Kiang. Neural networks. In Hossein Bidgoli, editor, *Encyclopedia of Information Systems*, pages 303–315. Elsevier, New York, 2003.

[73] Varacallo M Ludwig-PE, Reddy V. `https://www.ncbi.nlm.nih.gov/books/NBK441977/`. Updated: 07-24-2023.

[74] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952.

[75] E.R. Kandel. *In Search of Memory: The Emergence of a New Science of Mind*. W. W. Norton, 2006.

[76] P. Nelson. *Física biológica: Energía, información, vida*. Reverte, 2018.

[77] R.P. Feynman, R.B. Leighton, M. Sands, E.O. L, H.E. D, C.A. Heras, J.M. Marfil, and R. Gómez. *Lecciones de física de Feynman, I: Mecánica, radiación y calor*. Ediciones Científicas Universitarias. FCE - Fondo de Cultura Económica, 2019.

[78] L.F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5):303–304, 1999.

[79] P. Vlamos. *GeNeDis 2022: Computational Biology and Bioinformatics*. Advances in Experimental Medicine and Biology. Springer International Publishing, 2023.

[80] W. Gerstner, W.M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[81] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[82] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[83] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[84] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *ArXiv*, abs/1710.05941, 2018.

[85] Petr Dolezel, Pavel Skrabanek, and Lumir Gago. Weight initialization possibilities for feedforward neural network with linear saturated activation functions. *IFAC-PapersOnLine*, 49(25):49–54, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[86] Tomasz Szandala. *Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks*, pages 203–224. Springer Singapore, Singapore, 2021.

[87] Marc Artzrouni. Mathematical demography. In Kimberly Kempf-Leonard, editor, *Encyclopedia of Social Measurement*, pages 641–651. Elsevier, New York, 2005.

[88] W.H. Beyer. *Handbook of Mathematical Science*. CRC Press, 2018.

[89] John A. Drakopoulos. Sigmoidal theory. *Fuzzy Sets and Systems*, 76(3):349–363, 1995.

[90] G. Bebis and M. Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[91] Edmund T. Rolls and Alessandro Treves. 54Competitive networks, including self-organizing maps. In *Neural Networks and Brain Function*. Oxford University Press, 10 1997.

[92] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *ArXiv*, abs/1912.05911, 2019.

[93] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[94] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[96] S. M, P.K. K, V. S, and P.R. Podaralla. *Intelligent Computing: An Introduction to Artificial Intelligence*. Shineeks Publishers, 2023.

[97] David A. Elizondo and Emile Fiesler. A survey of partially connected neural networks. *International journal of neural systems*, 8 5-6:535–58, 1997.

[98] Pei-Chann Chang, Di di Wang, and Chang le Zhou. A novel model by evolving partially connected neural network for stock price trend forecasting. *Expert Systems with Applications*, 39(1):611–620, 2012.

[99] Smaranda Belciug and Elia El-Darzi. A partially connected neural network-based approach with application to breast cancer detection and recurrence. In *2010 5th IEEE International Conference Intelligent Systems*, pages 191–196, 2010.

[100] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[101] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[102] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[103] Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990.

[104] Kurt and Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[105] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 7:87–93, 2018.

[106] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

[107] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[108] James L McClelland and David E Rumelhart. *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. MIT press, 1989.

[109] F. Curatelli and O. Mayora-Ibarra. Competitive learning methods for efficient vector quantizations in a speech recognition environment. In Osvaldo Cairó, L. Enrique Sucar, and Francisco J. Cantu, editors, *MICAI 2000: Advances in Artificial Intelligence*, pages 108–114, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[110] Robert Cierniak and Leszek Rutkowski. On image compression by competitive neural networks and optimal linear predictors. *Signal Processing: Image Communication*, 15(6):559–565, 2000.

[111] Toshio Uchiyama and Michael A. Arbib. Color image segmentation using competitive learning. *IEEE Transactions on pattern analysis and machine intelligence*, 16(12):1197–1206, 1994.

[112] Ezequiel López-Rubio, Juan Miguel Ortiz-de Lazcano-Lobato, José Muñoz-Pérez, and José Antonio Gómez-Ruiz. Principal Components Analysis Competitive Learning. *Neural Computation*, 16(11):2459–2481, 11 2004.

[113] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.

[114] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342, 2014.

[115] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[116] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[117] Xindi Cai, Nian Zhang, Ganesh K. Venayagamoorthy, and Donald C. Wunsch. Time series prediction with recurrent neural networks trained by a hybrid pso–ea algorithm. *Neurocomputing*, 70(13):2342–2353, 2007. Selected papers from the 3rd International Conference on Development and Learning (ICDL 2004) Time series prediction competition: the CATS benchmark.

[118] Masayo Inoue, Mana Futamura, and Hirokazu Ninomiya. No one-hidden-layer neural network can represent multivariable functions. *ArXiv*, abs/2006.10977, 2020.

[119] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[120] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press.

[121] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*, 2023.

[122] Yoshua Bengio. *Practical Recommendations for Gradient-Based Training of Deep Architectures*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[123] Anna Rakitianskaia and Andries Engelbrecht. Measuring saturation in neural networks. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1423–1430, 2015.

[124] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[125] Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[127] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[128] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[129] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591 vol.1, 1993.

[130] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[131] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[132] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[133] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015. PMLR.

[134] Geoffrey I. Webb. *Overfitting*, pages 744–744. Springer US, Boston, MA, 2010.

[135] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2):022022, feb 2019.

[136] Reza Moradi, Reza Berangi, and Behrouz Minaei. A survey of regularization strategies for deep models. *Artificial Intelligence Review*, 53:3947–3986, 2020.

[137] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

[138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[139] Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[140] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer New York, 2013.

[141] P.I. Viñuela and I.M.G. León. *Redes de neuronas artificiales: un enfoque práctico*. Pearson Educación, 2004.

[142] M. Beckerman and J.P. Jones. Analysis of errors in a special perturbations satellite orbit propagator, February 1999.

[143] Tomasz Hadas and Jaroslaw Bosy. Igs rts precise orbits and clocks verification and quality degradation over time. *GPS Solutions*, 19:1–13, 01 2014.

[144] Han Cai, Yang Yang, Steve Gehly, Changyong He, and Moriba Jah. Sensor tasking for search and catalog maintenance of geosynchronous space objects. *Acta Astronautica*, 175, 06 2020.

[145] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374, 2016.

[146] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, 2016.

[147] P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer New York, 2009.

[148] Khaled H. Hamed. Trend detection in hydrologic data: The mann–kendall trend test under the scaling hypothesis. *Journal of Hydrology*, 349(3):350–363, 2008.

[149] A. C. Harvey. *ARIMA Models*, pages 22–24. Palgrave Macmillan UK, London, 1990.

[150] Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28, 1985.

[151] *Vector Autoregressive Models for Multivariate Time Series*, pages 385–429. Springer New York, New York, NY, 2006.

[152] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[153] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[154] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[155] Juan Félix San-Juan, Montserrat San-Martín, Iván Pérez, Rosario López, Eliseo P Vergara, Roberto Armellin, Martín Lara, Alexander Wittig, and Dario Izzo. Hybrid orbit propagation enhancement of the sgp4 propagator through machine-learning techniques. techreport 16-4101, European Space Agency, the Advanced Concepts Team, 2018. Available online at www.esa.int/act.

[156] Edna Segura, Hans Carrillo, Rosario López Gómez, Iván Pérez, Montserrat San-Martín, and Juan F San Juan. Deep learning hsgp4: Hyperparameters analysis. In *SPACEFLIGHT MECHANICS 2021: Proceedings of the 31th AAS/AIAA Space Flight Mechanics Meeting held February 1–3, 2021, Virtual Event*, volume 176, pages 1299–1313. Univelt Inc., 2021.

[157] Coşkun Hamzaçebi, Diyar Akay, and Fevzi Kutay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications*, 36:3839–3844, 03 2009.

[158] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[159] Spencer Aiello, Tom Kraljevic, and Petr Maj. *h2o: R Interface for H2O*. R Foundation for Statistical Computing, 2016. R package version 3.8.3.3.

[160] H2O user guide.

[161] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[162] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[163] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[164] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[165] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[166] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[167] François Chollet et al. Keras. `https://keras.io`, 2015.

[168] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat

Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[169] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[170] Montserrat San Martín Pérez. *Métodos de propagación híbridos aplicados al problema del satélite artificial: técnicas de suavizado exponencial*. PhD thesis, 2014.

[171] Stan Salvador and Philip Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. In *KDD workshop on mining temporal and sequential data*, volume 6, pages 70–80. Seattle, Washington, 2004.

[172] S.L. Mirtaheri and R. Shahbazian. *Machine Learning: Theory to Applications*. CRC Press, 2022.

[173] S. Bileschi, E. Nielsen, and S. Cai. *Deep Learning with JavaScript: Neural networks in TensorFlow.js*. Manning, 2020.

[174] John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. Parallel random numbers: As easy as 1, 2, 3. In *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2011.

[175] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[176] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[177] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[178] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[179] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[180] python-multiprocessing - pypi.

[181] python-subprocess - pypi.

[182] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[183] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.